COO-3077-139

Courant Institute of

Mathematical Sciences

ERDA Mathematics and Computing Laboratory

The Method of Lines Solution of Partial Differential Equations

James M. Hyman

ERDA Research and Development Report

Mathematics and Computing
October 1976

New York University

NEW YORK UNIVERSITY
NEW YORK UNIVERSITY
COURANT INSTITUTE-LIBRARY
251 Mercer St. New York, N.Y. 10012

UNCLASSIFIED

ERDA Mathematics and Computing Laboratory

Courant Institute of Mathematical Sciences
New York University

Mathematics and Computing COO-3077-139

THE METHOD OF LINES SOLUTION OF PARTIAL DIFFERENTIAL EQUATIONS

James M. Hyman

October 1976

Contract No. E(11-1)-3077

UNCLASSIFIED

Table of Contents

	Page
Abstract	iv
0. Introduction	1
1. Space Discretization	6
2. Time Discretization	24
3. Optimizing Space-Time Discretization	31
4. Description of MOL1D: A Method of Lines Subroutine Package	35
5. Numerical Examples	42
6. Discussion	80
Appendix	82
Bibliography	103

Abstract

In this paper we give an analysis of effective strategies for the so-called method of lines solution of the initial boundary value problem for partial differential equations of the form

$$u_t = g(x,t,u,u_x,u_{xx},f_x)$$

where

$$f = f*x,t,u,u_x,u_{xx}) \quad .$$

We also describe a user oriented FORTRAN subroutine package called MOL1D for the numerical solution of equations of this type using the method of lines.

Introduction

The method of lines (MOL) is a convenient, reliable technique to approximate the solution of initial value problems for systems of linear and nonlinear partial differential equations (PDEs). The method is capable of solving a wide variety of different types of PDEs and it has been successfully applied to such diverse equations as chemical reaction diffusion equations, systems of conservation laws, nonlinear wave equations, and nonlinear elliptic boundary value problems.

The initial value problem for a PDE can be written in operator form

$$u_t = g(u) , \quad u(0) = u_0 , \tag{0.1}$$

where $u(x,t)$ lies in some function space and g is a nonlinear discontinuous operator. The boundary conditions for the PDE are incorporated as restrictions on the domain of g.

In the MOL, u is approximated by an element U of some finite dimensional space whose components are the values of u at a discrete set of points. g is approximated by an operator G mapping that space into itself. The approximation to (0.1),

$$U_t = G(U) , \quad U(0) = U_0 , \tag{0.2}$$

is a system of ODEs, which is then integrated numerically.

There are two crucial aspects of MOL which determine the success of the method. The first involves a choice as to the method used to approximate the spatial derivatives and incorporate the boundary conditions into G. The second involves a choice as to the method used to solve the ODE (0.2) numerically.

The possible advantages of MOL are:

(a) By separating the problem of space and time discretization it is easy to establish stability and convergence for a wide range of problems. See Abrashin [1], Liskovets [22], Thompson [44], Voigt [46], and Walter [47].

(b) The powerful numerical techniques for solving ODEs, such as dynamically changing the step size and method to maintain stability and a desired time integration accuracy, can be directly applied to the PDE case.

(c) Programming effort can be substantially reduced by making use of well documented and reliable ODE packages.

(d) By solving the ODEs very accurately one can compare the accuracy and efficiency of different approximations of spatial derivatives.

A possible disadvantage in using MOL is the loss in the overall optimization of the method by decoupling the analysis of the space and time discretizations.

The organization of this paper is as follows:
In Section 1 we discuss various spatial discretizations including difference approximations, and Fourier methods. Our

guiding principle in the choice of discrete models is to retain as closely as possible all crucial properties of the PDE. For instance, for a hyperbolic PDE the operator g in (0.1) is antisymmetric; we endeavor to approximate g by an antisymmetric matrix G, at least with respect to all modes that are relevant. For a parabolic PDE the operator g is dissipative; we shall approximate g by a dissipative matrix G. When g is in conservation form, we shall choose a form for G which is also in conservation form.

We remark that an ODE of the form (0.2) where G is an antisymmetric or dissipative matrix is stable. Since stability implies convergence we conclude that solutions of (0.2) converge to those of (0.1). For more refined error estimates we rely on the literature of MOL, see [22] and [47].

In Section 1 we also discuss discrete models, paying particular attention to the efficiency of using higher order differences, mesh refinement and to the way boundary conditions are approximated. Many of our conclusions in this section are based on the work of Kreiss and Oliger [26]. The ideas are illustrated on the prototypes $u_t = c\, u_x$ and $u_t = u_{xx}$ of a hyperbolic and a parabolic equation respectively.

In Section 2 we review the efficiency of the numerical methods available for solving systems of ODEs in MOL calculations. We rely heavily on the studies of Hull, Enright, Fellen and Sedgwick [21] and Shampine, Watts and Davenport [38].

Since the operator g in equation(0.1) is unbounded, the matrix G which approximates g will have both small and large eigenvalues; this makes the ODE system (0.2) inevitably stiff. In the hyperbolic case this stiffness results in oscillating modes whose frequencies are very disparate; in the parabolic case it results in modes which decay at very different rates. For nonlinear problems, such as occur in chemical engineering, there can be, for a very short time, modes that increase at a very fast rate and the ODE system can be extremely stiff. Thus we are forced to use numerical methods which can deal with stiff systems; this makes the choice of an implicit method almost a necessity. We pay particular attention to multistep methods applied to large systems.

We describe the coupling of the spatial and temporal descritizations in Section 3 by analyzing the computational efficiency of various MOL strategies in solving simple hyperbolic and parabolic equations. The work of Swartz and Wendroff [41], [42] combined with numerical experiments helps to choose effective combinations of spatial and temporal methods.

In Section 4 we describe a FORTRAN subroutine package, called MOL1D, for the solution of initial-boundary value problems for systems of PDEs using the MOL. The package requires the user to describe the nature of his problem -- hyperbolic or parabolic, linear or nonlinear, smooth or discontinuous and the spatial discretization to be used; there-

upon the package selects an appropriate numerical method for integrating the resulting system of ODEs. The package allows the user the option to override this choice and to call for any one of a number of ODE solvers. The output includes a description of the numerical method used. We emphasize that the package can be used by a scientist who has had only minimal experience with computing, such as a quickie FORTRAN course.

Section 5 presents a variety of examples of numerical solutions computed by MOL1D which are then compared with analytical solutions. The efficiency and accuracy of various space and time discretizations are evaluated.

Section 6 is the conclusion which, not surprisingly, extols MOL1D as a reliable tool which enables users little versed in the art of solving PDEs to solve, efficiently, fairly complicated problems.

Acknowledgment

I would like to express my sincerest appreciation to Professor Peter D. Lax for his sure guidance and constructive criticism during this work.

1. Space Discretization

We shall first describe the MOL for two typical initial value problems. They are the unidirectional wave equation $u_t = cu_x$, and the diffusion equation $u_t = u_{xx}$. The simple wave equation exhibits many basic properties of linear hyperbolic equations. We will discuss different spatial derivative approximations, approximations of derivatives near the boundaries and mesh refinement techniques for both equations. We also formulate proper boundary conditions for systems of hyperbolic equations, and show how to incorporate them into the MOL approximation.

Consider the initial value problem with periodic boundary conditions on the unit interval

$$u_t = c\, u_x$$

(1.1)

$$u(x,0) = f(x) \ .$$

The solution of (1.1) is

(1.2) $$u(x,t) = f(x + ct) \ .$$

Since equation (1.1) can be solved exactly, we shall use it to study the effect of replacing spatial derivatives by various numerical approximations. In general, the solutions of difference approximations to (1.1) are not of the form (1.2) and the comparison between exact and approximate solutions is facilitated by the use of Fourier analysis.

Let N be a natural number, define $h = 1/(2N+1)$ and $x_k = kh$, $k = 0,1,\ldots,2N$. At these points $f(x)$ can be represented by the finite Fourier series

$$f(x_k) = \sum_{j=-N}^{N} \hat{f}_j \, e^{2\pi i j x_k}$$

where

$$\hat{f}_j = \frac{1}{2N+1} \sum_{k=0}^{2N} f(x_k) \, e^{-2\pi i j x_k}.$$

Then

$$u(x,t) = \sum_{j=-N}^{N} \hat{f}_j \, e^{2\pi i j (x+ct)} \tag{1.3}$$

is a solution of equation (1.1) which takes on the initial values at the grid points x_k.

To approximate equation (1.1) by the MOL, we begin by replacing the spatial derivatives with second order centered differences. This results in a system of ODEs

$$U_t = A_2 U \tag{1.4}$$

where

$$U = \begin{pmatrix} u(x_0,t) \\ \vdots \\ u(x_{2N},t) \end{pmatrix}$$

and

$$A_2 = \frac{c}{2h} \begin{pmatrix} 0 & 1 & 0 & 0 & \ldots & 0 & -1 \\ -1 & 0 & 1 & 0 & & & 0 \\ 0 & -1 & 0 & 1 & & & \cdot \\ \vdots & & & \cdot & \cdot & & \cdot \\ 0 & & & & -1 & 0 & 1 \\ 1 & 0 & \ldots & & & -1 & 0 \end{pmatrix}$$

Equation (1.4) can also be solved exactly and it is left to the reader to verify that the solution is

$$u(x_k,t) = \sum_{j=-N}^{N} \hat{f}_j \, e^{2\pi i j(x_k+c_j t)} \tag{1.5}$$

where

$$c_j = c \frac{\sin 2\pi jh}{2\pi jh} .$$

The eigenfunctions of the approximate solution (1.5) are the same as in the exact solution (1.3) but the temporal eigenvalues differ. The eigenvalues of A_2 arepurely imaginary, as they should be for a hyperbolic operator, and so all the errors occur in the phase of the approximate solution.

The phase error in the k^{th} mode is

$$e_2(k) = c2\pi kt\left(1 - \frac{\sin 2\pi kh}{2\pi kh}\right) \tag{1.6}$$

$$\doteq \frac{ct}{6h} (2\pi\ kh)^3 .$$

Observe that if the initial values consist of a single mode, i.e.

$$f(x) = e^{2\pi ikx} ,$$

then the maximum error of the approximate solution at the mesh points x_j is

$$\begin{aligned} \|U_j(t)-u(x_j,t)\|_\infty &= \|e^{2\pi ik(x_j+ct)} - e^{2\pi ik(x_j+c_k t)}\|_\infty \\ &= \|e^{2\pi ikx_j}(e^{2\pi ikct} - e^{2\pi ikc_k t})\|_\infty \\ &\doteq \|e^{2\pi ikx_j}(c-c_k)\ 2\pi ikt\|_\infty \\ &\doteq e_2(k) \end{aligned}$$

We define $M_2 = (kh)^{-1}$ as the number of mesh points per wave length. The error can be expressed as

$$e_2(k) \doteq \frac{(2\pi)^3}{6M_2^2} \text{ ckt} \quad .$$

The number of mesh points needed per wave length for a given phase error e is

$$M_2 \doteq 2\pi \sqrt{\frac{\pi}{3e} \text{ ckt}} \quad .$$

M_2 is dependent not only on the phase error but also on the square root of the computation time interval, the wave speed c, and the relevant modes of the initial data.

If fourth order centered differences are used to approximate u_x instead of second order differences the phase error in the kth mode is

$$(1.7) \quad e_4(k) = c\ 2\pi\ kt \left[1 - \frac{8 \sin 2\pi kh - \sin 4\pi kh}{12\pi\ kh}\right]$$

$$\doteq \frac{(2\pi)^5}{30M_4^4} \text{ ckt}$$

The number of mesh points needed per wave length for a given phase error e is

$$M_4 = 2\pi \left(\frac{\pi}{15e} \text{ ckt}\right)^{1/4}$$

When the initial values consist of a single mode then for a given phase error (or maximum error) the relationship between the number of mesh points needed per wave length is

$$M_2 = \frac{\sqrt{5}}{2\pi} M_4^2 \doteq 0.36\ M_4^2 \quad .$$

This relationship is independent of the specified phase error and the computation time. The fourth order method is clearly superior in any reasonable calculation. Example (5.1) in Section 5 compares the accuracy and computation times of these methods. These tests confirm the analytical relationships we have shown here.

For very small errors or long computation times the sixth order method is more efficient than the fourth order method. A major difficulty with the sixth order method is obtaining stable high order approximations to boundary conditions. Gustafsson [13] has shown that the derivative approximations near the boundary should be at most one order less than the internal accuracy. High order derivative approximations near the boundary have a tendency to be unstable and for this study we restrict ourselves to fourth and lower order finite difference methods.

If the problem is periodic then an excellent choice to approximate spatial derivatives is the Fourier method. In the Fourier method we interpolate the solution at time t by a trigonometric polynomial of degree N, i.e.

$$(1.8)\qquad u(x,t) \doteq \sum_{j=-N}^{N} \hat{u}_j(t)\, e^{2\pi i j x}, \quad x = x_k,\ k = 0,1,\ldots,2N$$

$$\hat{u}_j(t) = \frac{1}{2N+1} \sum_{k=0}^{2N} u(x_k,t)\, e^{-2\pi i j x_k}.$$

Differentiating (1.8) we have an approximation to the derivatives of $u(x_k,t)$,

$$(1.9)\quad c\left.\frac{\partial u(x,t)}{\partial x}\right|_{x=x_k} \doteq c \sum_{j=-N}^{N} 2\pi i j\, \hat{u}_j(t)\, e^{2\pi i j x_k}.$$

These operations are accomplished numerically by two fast Fourier transforms. We remark that the symmetry of finite Fourier transforms of real functions should be used to prevent instabilities due to aliasing.

Let A_F be the operator defined by the right side of equation (1.9). The solution to the system of ODEs

$$(1.10)\qquad U_t = A_F U$$

approximating equation (1.1) is

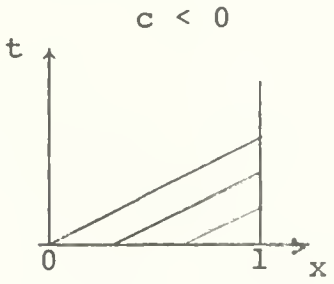
$$(1.11)\quad u(x_k,t) = \sum_{j=-N}^{N} \hat{f}_j\, e^{2\pi i j(x_k+ct)}.$$

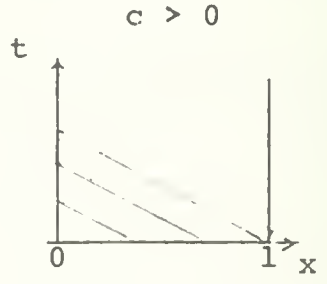
This is the exact solution.

There are drawbacks to the Fourier method. The FFT requires more operations per mesh point than either the second or fourth order difference methods. Also, the accuracy of finite Fourier series depends on the smoothness of the data. If the solution has sharp gradients or discontinuities, then many Fourier modes are necessary to approximate the solution accurately. This can occur for instance if a nonperiodic function is extended periodically in order to approximate

the derivatives with the FFT. The resulting function usually is discontinuous or has a discontinuity in a low order derivative at the old boundary. The resulting error then propagates inward and destroys the accuracy of the solution. For problems with nonperiodic boundaries there are generalizations of the Fourier, method such as the Galerkin or finite element collocation method, that give excellent high order approximations to the derivatives. We do not consider these generalizations here and refer the reader to a recent MOL implementation of the Galerkin method. (See Madsen and Sincovec [33].)

We now consider equation (1.1) with nonperiodic boundaries. We rely on the theory of initial boundary value problems for hyperbolic equations developed largely by Kreiss, see [24], [25]. The solution $u(x,t) = f(x + ct)$ is constant along the characteristics $x + ct = \text{constant}$.





The initial data uniquely determines the solution in the triangle $0 \leq x + ct \leq 1$. If $c > 0$ ($c < 0$) then a boundary condition $u(1,t) = g_1(t)$ $(u(0,t) = g_0(t))$ is needed to determine the solution outside the triangular region.

Furthermore, no boundary condition should be given at $x = 0$, $(x = 1)$.

Consider the hyperbolic system on the unit interval

$$\begin{aligned} v_t &= Hv_x + Kv \\ v(x,0) &= F(x) \end{aligned} \tag{1.12}$$

where H is a constant symmetric matrix. There exists a unitary matrix U such that

$$T = U^{-1}HU = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & \lambda_n \end{pmatrix}, \quad U^TU = I, \quad \lambda_i \text{ real.}$$

Define $w = U^{-1}v$ and arrange the transformed system such that

$$\begin{aligned} w_t &= Tw_x + U^{-1}KUw \\ w(x,0) &= U^{-1}F(x) \end{aligned}$$

or

$$\begin{pmatrix} w^I \\ w^{II} \end{pmatrix}_t = \begin{pmatrix} -T^I & 0 \\ 0 & T^{II} \end{pmatrix} \begin{pmatrix} w^I \\ w^{II} \end{pmatrix}_x + U^{-1}KU \begin{pmatrix} w^I \\ w^{II} \end{pmatrix} \tag{1.13}$$

where

$$T^I = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \\ 0 & \cdots & & \lambda_r \end{pmatrix}, \quad T^{II} = \begin{pmatrix} \lambda_{r+1} & 0 & \cdots & 0 \\ 0 & \lambda_{r+2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \\ 0 & \cdots & & \lambda_n \end{pmatrix}, \quad \lambda_i > 0,$$

and

$$w^I = \begin{pmatrix} w_1 \\ \vdots \\ w_r \end{pmatrix}, \qquad w^I = \begin{pmatrix} w_{r+1} \\ \vdots \\ w_n \end{pmatrix}.$$

w^I and w^{II} require boundary conditions at $x = 0$ and $x = 1$ respectively. We consider boundary conditions of the form

(1.14a) $w^I(0,t) = S^I w^{II}(0,t) + g^I(t)$

(1.14b) $w^{II}(1,t) = S^{II} w^I(1,t) + g^{II}(t)$.

S^I and S^{II} are matrices representing the dependence between w^I and w^{II} at the boundaries. The number of boundary conditions at $x = 0$ and $x = 1$ are the same for the transformed system as for the original equation. In particular if H has r negative eigenvalues and n-r positive eigenvalues then there must be r boundary conditions at $x = 0$ and n-r boundary conditions at $x = 1$.

For nonlinear hyperbolic systems boundary conditions should be prescribed as required by the corresponding linearized equations. This technique is used by Elvius and Sundström in [4] where they describe a collection of relevant well posed initial boundary value problems for the shallow water equations.

We now consider finite difference methods to incorporate the boundary conditions into the operator A of the semidiscrete PDE. The difference approximations near the boundary may be one order of accuracy lower than the approximations in the interior and still retain the overall higher accuracy. We are interested in fourth order difference methods and therefore we study third order approximations to boundary conditions.

Using five points we can get fourth order difference approximations to u_x and u_{xx}. If the five points are distributed symmetrically, then such a five point formula is applicable at all points except those one mesh away from the boundary,

and at the boundary. We rectify this situation by extending the solution to two mesh points beyond the boundary by cubic extrapolation. We then continue to use the five point centered differences at all the mesh points, substituting the fictitious points u_{-1}, u_{-2}, u_{n+1} and u_{N+2} where necessary at the left and right boundaries. The formulas for cubic extrapolation at the boundaries are

$$
\begin{aligned}
u_{-1} &= 4u_0 - 6u_1 + 4u_2 - u_3 , \\
u_{-2} &= 4u_{-1} - 6u_0 + 4u_1 - u_2 , \qquad \text{and} \\
u_{N+1} &= 4u_N - 6u_{N-1} + 4u_{N-2} - u_{N-3} , \\
u_{N+2} &= 4u_{N+1} - 6u_N + 4u_{N-1} - u_{N-2} .
\end{aligned}
\tag{1.15}
$$

This procedure is equivalent to using four point third order non-centered differences near the boundary.

The differential difference operators in the fourth order semi-discrete approximation

$$
U_t = A_4 U , \quad U(0) = U_0 ,
\tag{1.16}
$$

of the initial boundary value problem

$$
\begin{aligned}
u_t &= cu_x , \quad c > 0 , \quad 0 \leq x \leq 1 \\
u(x,0) &= f(x) \\
u(1,t) &= g(t)
\end{aligned}
\tag{1.17}
$$

is

$$A_4 = \frac{c}{12h}\begin{pmatrix} -22 & 36 & -18 & 4 & 0 & 0 & . & . & . & 0 \\ -4 & -6 & 12 & -2 & 0 & 0 & & & & . \\ 1 & -8 & 0 & 8 & -1 & 0 & & & & . \\ 0 & 1 & -8 & 0 & 8 & -1 & & & & . \\ . & & & & . & & & & & . \\ . & & & & & . & & & & . \\ . & & & & & & . & & & . \\ . & & & & & 1 & -8 & 0 & 8 & -1 \\ . & & & & & 0 & 2 & -12 & 6 & 4 \\ 0 & . & . & . & . & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$u(1,t)$ is defined by the boundary conditions rather than the differential equation but is included in U to simplify the notation.

We remark that when the boundary conditions involve derivatives with respect to t, they fit naturally into the framework of MOL. When the boundary conditions contain spatial derivatives such as

$$a(t)u + b(t)u_x = c(t) , \qquad x = x_0 \tag{1.18}$$

we proceed as follows. Replace u_x in (1.18) by the third order unsymmetric difference approximation

$$(u_0)_x = \frac{1}{6h}(-2u_{-1} - 3u_0 + 6u_1 - u_2) ,$$

and solve the resulting equation for u_{-1} ,

$$u_{-1} = \frac{3h}{b(t)}[a(t)u_0 - c(t)] - \frac{1}{2}(3u_0 - 6u_1 + u_2) .$$

The fictitious point u_{-2} is obtained by cubic extrapolation, see equation (1.15).

These procedures have proven effective for both hyperbolic and parabolic equations as is seen in examples (5.3) and (5.7) of Section 5.

The boundary conditions for systems of equations such as equations (1.14) are approximated by first defining the solution at the boundary for those components where a condition is given, and then extrapolating all the components using formulas (1.15). The PDE must be in characteristic form (1.13) to insure the accuracy in this procedure.

Another consideration in choosing a spatial discretization is that the solution of a PDE may be much smoother in some regions than others. For example there may be large gradients in a localized region such as a boundary layer. In such problems it is attractive to employ a local mesh refinement. This may be done by refining the grid in the original coordinate space or by a change of variables to stretch the regions where the solution has sharp gradients. The first method has the advantage that no extra formula manipulations need be done. The coordinate transformation method requires analytical calculations but has several advantages.

The numerical wave speed in approximations to hyperbolic equations depends upon the mesh size. If a wave is poorly represented on a coarse mesh and an abrupt change is made to a fine mesh, reflections can destroy the accuracy of the solution. (See Browning, Kreiss and Oliger [2].) By stretching the mesh with the help of a continuous change

of variables the mesh transition is necessarily continuous and reflections are less likely to cause trouble. Another advantage of the coordinate transformation method over mesh refinement is that the stability of the differential difference equations is obtained by standard techniques applied to the stretched PDE rather than using a specialized analysis in regions where the mesh is refined. Another advantage is that Fourier techniques and high order finite differences are easier to implement on an evenly spaced mesh. In Section 5 examples (5.4) and (5.5) clearly demonstrate the improvement in the resolution of solutions with sharp gradients by a coordinate transformation.

There are several instances when it is desirable to use a dissipative method for solving hyperbolic equations. The reasons are:

(i) Superfluous solutions with large phase errors can be controlled or wiped out by dissipative methods.

(ii) Dissipative methods are generally more stable and less likely to suffer from nonlinear instabilities than non-dissipative methods, see Fornberg [7].

(iii) The spread of errors due to inexact boundary approximations is reduced by dissipative methods, see Kreiss and Oliger [24].

(iv) When approximating equations with shocks it is necessary to add some form of dissipation to achieve proper entropy production, see Harten [14].

There are two ways to achieve this. We can add a controlled amount of dissipation to a nondissipative scheme as a high order viscous term or we can use a method which is inherently dissipative. Both methods are easy to implement in MOL. An advantage of adding a controlled amount of dissipation is that it can be restricted to small regions of the solution where it is needed, such as in a neighborhood of a shock or near the boundaries where instabilities are most likely to occur. This maintains the higher accuracy in the greater portion of the calculation where the artificial dissipation is not needed. A disadvantage of this method is the extra computational cost to evaluate the additional term in the equations. Methods which are dissipative by nature do not need any extra computation.

The disadvantage of automatic dissipative methods is the lack of control over the amount of dissipation, which depends on the mesh size. The accuracy of computations on a coarse mesh may be destroyed by too much dissipation. This is not always the case as should be noted by the highly successful dissipative Lax-Wendroff scheme. There are many instances when the built-in dissipation automatically delivers sufficient damping just where it is needed. Examples of dissipative methods in MOL calculations will be given in Section 5.

We now turn to a linear parabolic equation on the unit interval,

$$u_t = u_{xx}$$

(1.19)

$$u(x,0) = f(x)$$

We start with the space periodic case and represent f(x) as a trigonometric polynomial

$$f(x) = \sum_{j=-N}^{N} \hat{f}_j \, e^{2\pi i j x} .$$

Then the exact solution of (1.19) is the finite sum

$$u(x,t) = \sum_{j=-N}^{N} \hat{f}_j \, e^{2\pi i j x} \, e^{-4\pi^2 j^2 t} . \tag{1.20}$$

When we replace the spatial derivatives by second order centered differences we obtain the differential difference equation

$$U_t = B_2 U . \tag{1.21}$$

U is a vector of the values of u at the mesh points $x_k = kh$, $h = 1/(2N+1)$. B_2 is the symmetric matrix

$$B_2 = \frac{1}{h^2} \begin{pmatrix} -2 & 1 & 0 & . & . & . & 1 \\ 1 & -2 & 1 & 0 & . & . & . \\ 0 & 1 & -2 & 1 & 0 & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 1 & . & . & . & 0 & 1 & -2 \end{pmatrix} .$$

The exact solution to (1.21) is

$$U_k(t) = \sum_{j=-N}^{N} \hat{f}_j \, e^{2\pi i j x_k} \, e^{\frac{2}{h^2}(\cos 2\pi j h - 1)t} \tag{1.22}$$

Comparing the approximate solution (1.22) with the exact solution (1.20) we note that there is no phase error but there is an amplitude error; the approximate solution decays at the wrong rate. The error in the decay rate of the kth mode is

$$(1.23)\quad d_2(k) = \left[-4\pi^2k^2 - \frac{2}{h^2}(\cos 2\pi kh - 1)\right]t$$
$$\doteq \frac{4}{3}\pi^4k^4h^2t \ .$$

$d_2(k)$ is positive and therefore the solution of the differential difference equation decays more slowly than the solution of the PDE.

If the initial values are of the form

$$f(x) = c\, e^{2\pi ikx} \ ,$$

then the relative error of the approximate solution at the grid points x_j is

$$\text{relative error} = \frac{U_j(t) - u(x_j,t)}{u(x_j,t)}$$
$$= e^{\left[-4\pi^2k^2 - \frac{2}{h^2}(\cos 2\pi kh - 1)\right]t} - 1$$
$$= e^{d_2(k)} - 1$$
$$\doteq d_2(k)$$

Define $M_2 = (kh)^{-1}$ as the number of mesh points per wave length. Then the error in the decay rate is

$$d_2(k) = \frac{\pi^2(2\pi k)^2}{3M_2^2}\, t$$

The number of mesh points needed per wave length for a given relative error d is

$$M_2 \doteq \pi \left[\frac{(2\pi k)^2 t}{3d} \right]^{1/2} .$$

Similarly, for fourth order centered differences, we have

$$(1.24) \quad d_4(k) = \left[-4\pi^2 k^2 - \frac{1}{6h^2} (16 \cos 2\pi kh - \cos 4\pi kh - 15)\right]$$

$$\doteq \frac{32}{45} \pi^6 k^6 h^4 t$$

$$\doteq \frac{8\pi^4 (2\pi k)^2}{45 M_4^4} t$$

and

$$M_4 \doteq \pi \left[\frac{8 (2\pi k)^2 t}{45d} \right]^{1/4} .$$

Observe that if our initial conditions consist of a single mode the relationship between the number of mesh points per wave length needed for the same relative error is

$$(1.25) \qquad M_2 = \frac{\sqrt{30}}{4\pi} M_4^2 \doteq 0.44 \; M_4^2 .$$

This relationship is independent of the computation time or the relevent modes in the initial data.

Equations (1.23) and (1.24), containing the formulas for the eigenvalues of B_2 and B_4, show that the condition number of these matrices is proportional to the square of the number of mesh points. So the fewer the mesh points, the less stiff the ODEs we have to solve. A fourth order

method needs fewer mesh points; therefore the ODEs are much less tiff and hence easier to solve than in a second order calculation of the same accuracy.

When the boundary conditions are not periodic the same procedures used in the hyperbolic case also work for parabolic equations. We remarked earlier that for hyperbolic equations it is very hard to find high order approximations to boundary conditions which are stable. In contrast parabolic equations are much more stable, and therefore sixth order differences are an attractive possibility tc improve the accuracy and/or reduce the stiffness of the differential difference operator. (See Varah [50].)

When the boundary conditions are of the form (1.18) the difference operator B does not retain the symmetry present in the differential operator. Hicks and Wei [16] have shown in a number of cases that even though the difference operator is not symmetric it has negative real eigenvalues.

For spatially periodic problems the Fourier method again is an excellent choice and will solve equation (1.19) exactly for the first 2N modes. In Section 5,example (5.2) compares the computation times and accuracy of several difference schemes. These tests confirm that one should use at least fourth order differences whenever possible.

2. Time Discretization

In this section we review ODE techniques emphasizing those which are appropriate for large systems of the form

$$y' = f(t,y) , \quad y(0) = y_0 , \tag{2.1}$$

arising from semi-discrete approximations of PDEs. Our policy will be to use robust methods, capable of solving a large class of problems rather than specific methods, optimal for a particular problem. The ODE method we select will be implemented as part of an MOL package described in Section 4. We therefore consider methods available in codes whose reliability and efficiency are well established.

As we saw in Section 1, semidiscrete PDEs are necessarily stiff. The condition number of the differential difference operator is typically $O(N)$ for hyperbolic equations and $O(N^2)$ for parabolic equations, where N is the number of mesh points. The situation may be more extreme for nonlinear equations where the time constants can range over many orders of magnitude and the resulting ODEs can be extremely stiff. Therefore we must select integration methods capable of handling both mildly stiff and very stiff equations. There are many ODE methods in this class, see Willoughby [48] for example. However, it is generally accepted that the best methods include Runge-Kutta, extrapolation and the Adams multistep methods.

In selecting an integration method, we rely heavily on the excellent tests of ODE codes by Hull [19], Hull, Enright, Fellen, and Sedgwick [21] and Shampine, Watts and Davenport [38]. These tests compare the capabilities and efficiencies of some of the best ODE codes available. The codes were evaluated on how well they solved a wide class of problems under a variety of accuracy requirements. Their efficiency was determined by the computer time, storage, overhead and derivative evaluations used by each code.

These authors found that the Adams multistep methods are more efficient than either the Runge-Kutta or extrapolation methods under almost all accuracy requirements. One might have believed that Runge-Kutta would be a good choice for MOL calculations since it uses less storage than the multistep methods. However Runge-Kutta methods are prohibitively expensive to use when either derivative evaluations are costly or high accuracy is desired. The extrapolation methods are not optimal for MOL calculations since they do not integrate stiff equations as well as multistep methods and they require an enormous amount of storage.

There are several excellent ODE codes available containing multistep methods for both mildly stiff and very stiff problems. These codes include DVDQ [27], GEAR [17], GEARB [18], and STEP [37]. The GEARB code programmed by A. Hindmarsh was chosen for this study because it includes a good implementation of both the Adams-Moulton and Gear's stiffly stable multistep methods. The Adams-Moulton methods have a smaller truncation

error than Gear's methods and are more efficient when the equations are only mildly stiff. This occurs in MOL approximations of linear hyperbolic equations. The Gear methods are more economical when the equations are very stiff such as in chemical reaction-diffusion equations. It is very easy to switch methods in the GEARB package allowing the user to pick the better ODE method for his particular problem.

We now give a brief description of the Gear and Adams multistep methods. These methods are based on approximating the following identity for the solution of equation (2.1):

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(s,y(s))\, ds \,. \tag{2.2}$$

The integral is approximated by interpolating f with a polynomial determined by Taylor series and the values of f and y at past nodes. This results in an equation of the form

$$y_{n+1} = \sum_{i=0}^{k_1} a_i y_{n-i} + h \sum_{i=-1}^{k_2} b_i\, f(t_{n-i}, y_{n-i}) \,. \tag{2.3}$$

where $h = t_{i+1} - t_i$. In the case of the Adams-Moulton method of order p, $k_1 = 0$ and $k_2 = p-2$. In the case of Gear's method of order p, $k_1 = p-1$ and $k_2 = -1$. In both cases $b_{-1} \neq 0$ and equation (2.3) is implicit.

A solution of (2.3) satisfies

$$r(y_{n+1}) = y_{n+1} - \sum_{i=0}^{k_1} a_i y_{n-i} - h \sum_{i=-1}^{k_2} b_i\, f(t_{n-i}, y_{n-i}) = 0$$

and can be found by Newton iteration

$$y_{n+1}^{(j+1)} = y_{n+1}^{(j)} - \left(P_{n+1}^{(j)}\right)^{-1} r\left(y_{n+1}^{(j)}\right) \tag{2.4}$$

where

$$P_{n+1}^{(j)} = \frac{\partial r}{\partial y_{n+1}^{(j)}} = I - h\, b_{-1} \left.\frac{\partial f}{\partial y}\right|_{y_{n+1}^{(j)}} = I - h\, b_{-1} J_{n+1}^{(j)}$$

At each iteration we must update the Jacobian matrix $J = f_y$ and invert P. In large systems of ODEs this is an expensive operation. The computation effort can be reduced by approximating P^{-1} since this only affects the rate of convergence of the iteration and not the limiting values. It is often less costly to replace the matrix $(P_{n+1}^{(j)})^{-1}$ by $(P_{n+1}^{(0)})^{-1}$ or $(P_m^{(0)})^{-1}$ for some $m < n + 1$ without adversely affecting the convergence. If we take P = I, equation (2.4) reduces to function iteration and will converge if the spectral radius of the right-hand side is less than unity. This is so if h is chosen small enough; how small is governed by the eigenvalues of J. If the differential equations are stiff then function iteration generally requires a much smaller value of h than we are willing to use.

We now describe efficient methods to approximate P and P^{-1}. We are forced to approximate P rather than define it analytically since the Jacobian J is very complicated in ODEs arising from discretized systems of nonlinear PDEs. To do this we use finite differences. The (i,j)-th element of J can be approximated by

$$[f_i(y + \delta e_j) - f_i(y)]/\delta$$

for some small increment δ where e_j is the j^{th} standard unit vector. In differential difference equations J is sparse and it is important that the ODEs be ordered such that most of the nonzero elements are along the main diagonal. The multistep method can then take advantage of the sparseness by calculating P^{-1} with a banded version of the LU decomposition method. The GEARB does this rather than performing an explicit matrix inversion. A rougher estimate of P^{-1} can be made by approximating the directional derivative of f and using the diagonal approximation

$$P_{n+1} = I - h\, b_{-1} D_{n+1} .$$

D_{n+1} is a diagonal matrix with ith element

$$d_i = [f_i(t_{n+1}, y_{n+1}^{(0)} + \varepsilon) - f_i(t_{n+1}, y_{n+1}^{(0)})]/\varepsilon_i$$

$$\varepsilon = (\varepsilon_i)_1^N = \delta r(y_{n+1}^{(0)})$$

for some small increment $\delta < 0$. The diagonal of D_{n+1} is an approximation to the directional derivative of f taken in the same direction as the change $y_{n+1}^{(1)} - y_{n+1}^{(0)}$ would be if function iteration were used. Inaccurate Jacobian approximations usually result in forcing the package to take smaller time steps and iterate equation (2.4) more often at each time step. They do not affect the accuracy of the computed solution unless the solution is so ill behaved that the

iteration does not converge or even worse, converges to the wrong solution. (See example (5.8) in Section 5.)

A major advantage of multistep methods is the ease with which step size and the order of the method can be changed to achieve an efficient time discretization while maintaining a given accuracy. These changes are carried out automatically by the package through a decision process. In the GEARB package, after equation (2.3) has been solved for y_{n+1} an estimate of the truncation error is made by a backward difference scheme using y_{n+1}. If we are currently using a p-th order method with time step h, the relative error is of the form

$$\varepsilon_p(n) = \frac{C_p h^{p+1}}{\|y\|} \frac{d^{p+1} y_{n+1}}{dt^{p+1}} . \tag{2.5}$$

The constant C_p depends upon the difference scheme being used. In MOL calculations the error is measured relative to the norm of the PDE solution rather than the ODE. If this error is less than the specified error tolerance, then the value y_{n+1} is accepted. If the error test fails, then the step is repeated with a smaller h and/or a lower order method until the error test is passed. Every (p+1)-st step the maximum h which could be used in the integration formula of order p-1, p and p+1 is estimated by solving equation (2.5) for h. The method chosen for the next time step is the one with the largest time step. Shampine and Gordon [37] describe an improved version of this procedure by using local extrapolation. At the stage

where the error has been estimated, it is subtracted from the solution. This not only improves the accuracy of the approximation, but it increases the stability of the numerical method.

Multistep methods use few function evaluations to solve a typical problem. This is done at the expense of using more storage. In one dimensional MOL problems, storage is not usually a major factor in choosing a method. However, this is not the case in higher dimensions. The storage required by the ODE package can be reduced by using some special properties of semidiscrete PDEs. For example, there is usually no point in striving for smaller errors in the time integration than what already is present due to the errors in the space discretization. Therefore, there is no reason to use very high order ODE methods. By restricting the package to, say, fifth order, storage can be saved without a loss of efficiency.

3. Optimizing the Space-Time Discretization

In this section we consider methods of choosing efficient combinations of spatial and temporal discretizations to achieve a given level of accuracy. To do this we rely on the results of Swartz and Wendroff [41], [42] for the equations $u_t = u_x$ and $u_t = u_{xx}$. In MOL approximations to first order there are two types of errors:

$$E_s = \text{spatial trunction error,}$$

and

$$E_t = \text{temporal trunction error.}$$

The total truncation error, E, for a scheme is often approximated by

$$E = E_s + E_t = C_p(\Delta x)^p + C_q(\Delta t)^q \tag{3.1}$$

with the coefficients independent of x and t. Furthermore, the computational effort or work, W, is approximated by

$$W = tC_W/(\Delta x\ \Delta t) \tag{3.2}$$

in one dimensional problems. C_W is a constant depending on the discretizations used and t is the length of integration. For a given accuracy the minimum work occurs when the errors are balanced such that

$$p\ C_p(\Delta x)^p = q\ C_q(\Delta t)^q\ . \tag{3.3}$$

The coefficients C_p and C_q can be approximated with the methods described in Sections 1 and 2. Using these approxi-

mations the minimal work for p-th order spatial and q-th order temporal scheme is

$$(3.4)\quad W_p^q = C_W (C_p/q)^{1/p}\ (C_q/p)^{1/q}\ [(p+q)/E]^{1/p + 1/q}\ .$$

Equation (3.4) can be used to measure the efficiency of the MOL scheme. The asymptotic work

$$W_p^q \sim E^{-(1/p + 1/q)}$$

for some standard space and time method combinations are

Space Accuracy p	Time Accuracy q	Asymptotic Work W
1	1	E^{-2}
2	1	$E^{-3/2}$
2	2	E^{-1}
4	2	$E^{-3/4}$
4	4	$E^{-1/2}$
6	4	$E^{-5/12}$

When high accuracy is desired, this table clearly illustrates the superiority of the fourth and sixth order schemes over the first and second order schemes.

These results of Swartz and Wendroff and the numerical tests presented in Section 5 imply that, in general, low order methods in either space or time are not competitive. If only crude accuracy (∼1% error) is needed then second order methods

are comparable to the higher order methods. However, when high accuracy ($\sim$0.01% error) is desired then the higher order methods are far more efficient. The tests in Section 5 compare the computer running time and accuracies of many different combinations of MOL integration methods on a diverse collection of problems. The greater efficiency of high order methods is evident time and time again in these tests. These relationships were also arrived at independently for the spatial and temporal discretizations in Sections 1 and 2 and are not surprising for the total method.

Minimizing the work depends upon being able to balance the spatial and temporal errors. Even though the results of Swartz and Wendroff cannot be applied directly to nonlinear systems of PDEs, we might expect some general trends to carry over. We attempt this by balancing the space and time errors in equation (3.3) numerically. An estimate of the spatial accuracy can be found by evaluating the spatial operator very accurately by employing a high order difference scheme at selected times during the calculation. For the temporal accuracy control we note that relation (3.3) minimizes the overall work with respect to Δt for a given allowable error. In an ODE package we have at our disposal not only the choice Δt but also q, the accuracy of the method used. Rather than performing a minimization with respect to Δt and q we recommend that the internal accuracy parameter of the ODE solver be set to the estimated spatial error to obtain balanced accuracy in the total discretization.

We conclude this section with a warning that balancing the spatial and temporal errors does not always result in an optimal difference scheme. There are instances when specifying a smaller error bound for the ODE solver than called for by this simple analysis will reduce the total computation time. The tighter error control can prevent instabilities in the discrete space-discrete time difference equations from creeping into the calculations when these instabilities are not present in the discrete space-continuous time differential equations. (See example (5.8) in Section 5.) The best rule is to experiment with different temporal accuracies if a particular equation is to be solved many times.

4. Description of MOL1D: A Method of Lines Subroutine Package

In this section we describe a FORTRAN subroutine package, called MOL1D, for the method of lines solution of systems of PDEs in one space dimension. MOL1D was programmed with two goals in mind. First, the package is designed so that a scientist can proceed directly to computing acceptable solutions to PDEs. This is done by eliminating many of the redundant programming efforts in preparing PDE programs. Substantial effort has been made to make the software reliable, efficient and convenient to use. Thoroughly tested numerical algorithms are conveniently available to users with a minimum of computing experience. This allows the user to solve a variation of the original problem with only slight programming changes.

Second, MOL1D is a useful tool for the numerical analyst in comparing different temporal and spatial discretization techniques. For example, the accuracy of different spatial approximations can be compared by accurately integrating the time derivative. The package offers many combinations of methods and is structured so more can be easily added.

There are a few generalized software packages for the numerical solution of PDEs already available. These include DCG [5], DSS [49], LEANS [36], PDECOL [33], PDEL [3], PDELAN [15], and PDEPACK [32]. Most of these codes

are not suitable for general purposes since they are not able to handle a wide range of PDEs. The exceptions to this are the two excellent MOL packages PDEPACK and PDECOL developed by N. Madsen and R. Sincovec. Their packages use respectively finite differences and finite element collocation methods to approximate the spatial derivatives and a modified version of the GEARB package to integrate the time variable. Although they were designed primarily for chemical reaction diffusion equations, they can be used on a wide assortment of problems. The program structure of PDEPACK and PDECOL was very helpful in designing MOL1D. MOL1D was programmed to allow experimentation with the MOL techniques on a very large class of one dimensional PDEs and is more general than any of the above programs.

The systems of PDEs numerically integrated by MOL1D must be well posed and of the form

$$(4.1)\quad u_t = g(x,t,u,u_x,u_{xx},f_x)\ ,\quad f = f(x,t,u,u_x,u_{xx})\ ,$$

on the finite interval $[x_0,x_N]$. u is a vector of dimension NPDE and f and g are vector valued functions. The package requires initial values for u and accepts both periodic boundaries and boundary conditions of the form

$$au + bu_x = c\ .$$

If $b \neq 0$, then a, b and c are functions of t and u. Otherwise, a and c are functions of t only. MOL1D then computes a numeri-

cal approximation to the solution of (4.1) at values of the independent variable $t \in [t_0,T]$ as desired by the user. The user has complete control over a large selection temporal and spatial derivative approximations available within the package. The time variable is integrated with the GEARB package using either Gear's or the Adams-Moulton multistep method. The spatial derivatives are approximated by second or fourth order symmetric difference, second or third order unsymmetric differences or by the Fourier method. Unsymmetric differences are restricted to hyperbolic equations in characteristic form (see equation (1.13)). The Fourier method may be used only in periodic problems. In nonperiodic problems the third order unsymmetric spatial derivative approximation described in Section 1 is used at the mesh points neighboring the boundaries.

To use MOL1D the programmer must provide a user written main program and three subroutines. The algorithm parameters are set in the main program. The three subroutines supplied by the user are: subroutine PDE to define u_t in terms of g and f_x as in equation (4.1), subroutine FUNC to define f, and subroutine BNDRY to define a, b, and c in equation (4.2). MOL1D is called by the main program. It then forms and integrates a semidiscrete approximation to the PDE. All intermediate output is provided by the package and control is returned to the main program only after the completion of the problem.

The MOL1D package is composed of twenty Fortran subroutines including a modified version of the GEARB ODE package. Figure 4.1 shows the lines of communication between these subroutines and the routines supplied by the user. The subroutines in MOL1D have the following names and perform the indicated tasks.

MOL1D	Driver routine for the package. It allocates storage, performs initialization tasks and monitors the progress of the time integration.
CKINP	Checks the validity of the input parameters
SOLPRT	Prints the solution at the user prescribed times.
SOLPLT	Plots the solution at the user prescribed times.
SOLDMP	Prints all relevant space derivatives of U and error estimates for the temporal integration. Used for debugging and if a failure flag is returned from GEARB.
DIFFUN	Defines the time derivatives of U for integration package GEARB.
BEVAL	Evaluates the boundary conditions and incorporates them into the equation by cubic extrapolation to fictitious points outside the region of integration.
UDER	Evaluates the derivatives of U with respect to x.
FDER	Evaluates the derivatives of f with respect to x.
FFT842	A fast Fourier transform routine
FFTFILT	Filters high modes of the discrete Fourier transform.
COMPARE	The user may call this routine to compute the accuracy in calculations when an exact solution is available.

DRIVEB Driver routine for the GEARB integration package. Other routines in GEARB include INTERP, STIFFB, COSET, PSETB, DECB, SOLB, and PDB.

The graphics option in MOL1D is a deviation from the general portability policy used in programming the package. The author believes that graphical output is essential to understanding the behavior of even relatively simple PDEs. All plotting procedures use standard CALCOMP routines and sufficient documentation is available within the package to modify these routines to other systems.

In many problems we can estimate the accuracy of our calculations by solving a similar equation where an exact solution is known. The accuracy in these calculations can be computed in subroutine COMPARE. This subroutine measures the difference between the exact solution, u, and the computed solution, v, in the L_1, L_2 and L_∞ norms. In periodic problems where the solution consists of a single mode the phase difference between u and v is computed and scaled to the range $[-\pi,\pi]$. v is then translated to eliminate the phase error and the distortion of the computed solution is measured by computing the difference between u and the translated computed solution.

A detailed description of the calling sequence and the arguments of the package and the routines supplied by the user is contained in the Appendix.

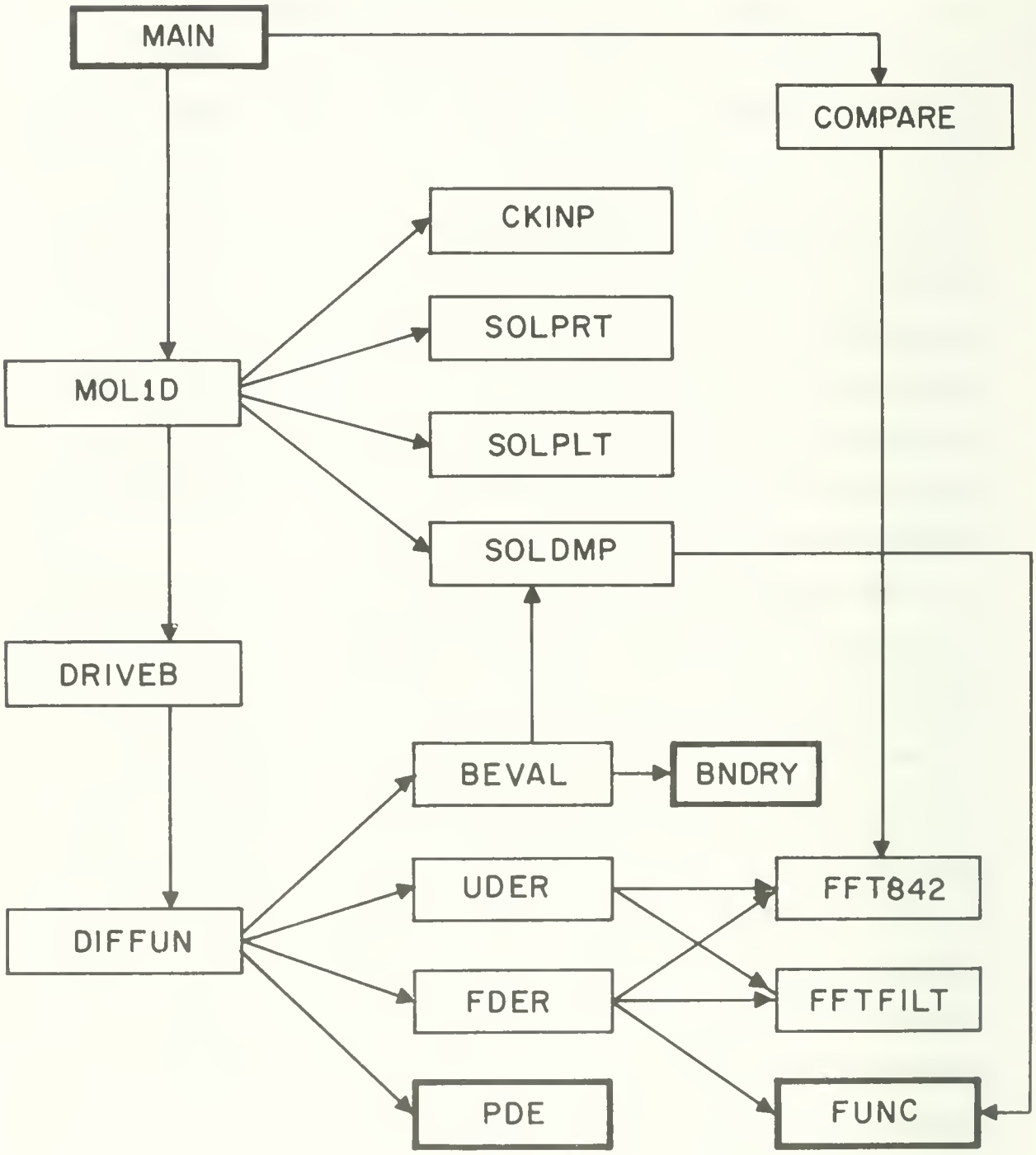


Figure 4.1. The Direction of the Procedure Calls in a MOL1D Operational Program. Routines Supplied by the User are Indicated by Heavy Boarders.

A user's guide will be available in the near future. At present the package is undergoing evolutionary changes at the Courant Institute and should not be considered to be stabilized as general purpose software. The package is not available for distribution at this time (Fall 1976), since the author believes that more harm than good would be accomplished by a premature release of the package.

5. Numerical Examples

In this section we present eight examples of linear and nonlinear PDEs that have been solved numerically with the MOL1D package. The computing times and accuracies of various space and time discretizations are compared. These examples test the reliability and efficiency of the package and help find its capabilities and limitations. Table V.1 lists the equations, initial conditions, boundary conditions and solutions of the problems that will be studied.

The following is a list of terms appearing in the tables of this section.

CPU time = total computer time used by MOL1D to integrate the equation on the CDC 6600 at the Courant Inst.

PDE calls = the number of evaluations of the right-hand side of the PDE.

Average temporal order = average order of the integration used at each time step

EPS = accuracy control parameter for local truncation error in integrating the differential difference equations

Phase error = the error in the dominant mode of the Fourier transform of the computed solution scaled to the range $[-\pi,\pi]$.

L_2 error = discrete L_2 deviation of the exact solution from the numerical solution

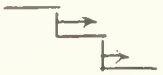
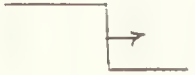
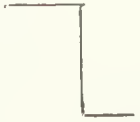
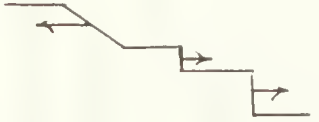
Corrected L_2 error = the discrete L_2 deviation of the exact solution from the numerical solution shifted to correct for the phase error.

The FORTRAN floating point notation is used to represent numbers, e.g. 5.3E-2 = 5.3×10^{-2}.

TABLE V.1

Example	Equation	Init. Conditions	Boundary Conditions	Solution	#(N)/ Mesh Points	Temporal Integration Method
1a	$u_t = u_x$ $-\pi \leq x \leq \pi$	sin x	periodic	sin x + t	$8 \leq N \leq 64$	Variable
1b	$u_t = u_x$ $-\pi \leq x \leq \pi$	sin kx k=1,2, ...8	periodic	sin k(x + t)	32	Variable
2	$u_t = u_{xx}$ $-\pi \leq x \leq \pi$	sin x	periodic	e^{-t} sin x	$16 \leq N \leq 128$	Variable
3a	$u_t = u_x$ $-\pi \leq x \leq \pi$	2+ cos x	$u(\pi,t) = 1$	$\begin{cases} 2+\cos x+t, & x+t<\pi \\ 1, & x+t \geq \pi \end{cases}$	$16 \leq N \leq 64$	Adams-Moulton Banded Jacobian EPS = 10^{-6}
3b	$u_t = u_x$ $-\pi \leq x \leq \pi$	1	$u(\pi,t) = 2 + \cos t$	$\begin{cases} 1, & x+t<\pi \\ 2+\cos x+t, & x+t \geq \pi \end{cases}$	$16 \leq N \leq 64$	Adams-Moulton Banded Jacobian EPS = 10^{-6}
3c	$u_t = u_x$ $-\pi \leq x \leq \pi$	1	$u_t(\pi,t) = -\sin t$	$\begin{cases} 1, & x+t<\pi \\ 2+\cos x+t, & x+t \geq \pi \end{cases}$	$16 \leq N \leq 64$	Adams-Moulton Banded Jacobian EPS = 10^{-6}
3d	$u_t = u_x + u_{xx}$ $-\pi \leq x \leq \pi$	sin x	$u + u_x = -e^{-t}(\sin t+\cos t)$ $x = \pm \pi$	e^{-t} sin x + t	$16 \leq N \leq 64$	Adams-Moulton Banded Jacobian EPS = 10^{-6}
3e	$u_t + \cos t\, u_x = \cos(x-t) \cdot [\cos t - 1]$	sin x	$u(\pi,t) = -\sin t$ if $\cos t \geq 0$ $u(-\pi\ t) = -\sin t$ if $\cos t \leq 0$	sin (x - t)	$16 \leq N \leq 64$	Adams-Moulton Banded Jacobian EPS = 10^{-6}
3f	$u_t = 3u_x - 4v_x$ $v_t = 2u_x - 3v_x$	u=3sinx v=2sinx	u-2v=sin t, x= 0 u- v=sin t, x= 2π	u= 2 sin(x+t)+ +sin (x-t) v=sin(x+t)+sin(x-t)	$16 \leq N \leq 64$	Adams-Moulton Banded Jacobian EPS = 10^{-6}

TABLE V.1
(Continued)

Example	Equation	Initial Conditions	Boundary Conditions	Solution	# of Mesh Points	Temporal Integration Method
4	$u_t = u_x + 0.003u_{xx}$ $0 \leq x \leq 1$	$2 + \cos \pi(2x-1)$	$u(0,t) = 1$ $u(1,t) = 1$		64	Adams-Moulton Banded Jacobian EPS = 10^{-6}
5	$u_t = u_{xx} - n \sinh n u$ $0 \leq x \leq 1$	x	$u(0,t) = 0$ $u(1,t) = 1$		64	Gear's Method Banded Jacobian EPS = 10^{-6}
6	$u_t = uu_x + 0.003u_{xx}$		consistent with exact solution		101, 201	Gear's Method Banded Jacobian EPS = 10^{-5}
7	Euler equations of gas dynamics		reflecting boundaries		81	Adams-Moulton Function iteration EPS = 10^{-4}
8	Chemical diffusion reaction equations		zero flux condition		8	Variable

Example 5.1a is the unidirectional wave equation

$$u_t = u_x \,, \qquad -\pi \leq x \leq \pi$$
$$u(x,0) = \sin x \tag{5.1}$$

with periodic boundaries.

The exact solution is

$$u(x,t) = \sin (x+t) \quad .$$

This equation was analyzed in Section 1 and many of the results there are well illustrated by this example.

Table V.2 shows the results of the MOL solution of (5.1) at time $t = 2\pi$ using fourth order spacial differencing as described in Section 1 in combination with various temporal integration methods as described in Section 2. Notice that decreasing the temporal error from 10^{-4} to 10^{-6} does not decrease the overall phase error in the case of 8 or 16 mesh points. The reason is that the space error dominates on such a crude mesh. On the other hand, in the case of 32 or 64 mesh points the overall error decreases when the temporal error is reduced from 10^{-4} to 10^{-6}.

For a fixed spatial mesh the CPU times for the methods using a diagonal approximation are progressively worse as the number of mesh points increases and the differential difference equations increase their stiffness. This trend is found in all the examples of this section and therefore we do not recommend using this option of the GEARB package for

MOL calculations. The banded approximations of the Jacobian require more storage than function iteration and are only necessary when the difference equations are very stiff which is not the case in this example.

Table V.3 contains results when the temporal method is fixed as the Adams-Moulton method with function iteration and the spatial difference methods are varied. The time variable was integrated very accurately (EPS = 10^{-6}) to isolate the errors due to the spatial discretization. The phase error of the Fourier method is approximately the same as the temporal accuracy since all modes of the solution are accounted for. The unsymmetric difference schemes are included in the package because of their usefulness automatically adding dissipation in shock calculations. They offer no advantage over either symmetric differences or the Fourier method in this example. As the table shows they are more expensive to use since the direction of the characteristics must be estimated at the mesh points. Notice that the shape of the solution is distorted by the unsymmetric differences as can be seen by the significant L_2 errors. These tests indicate that the Adams Moulton method with function iteration used in combination with fourth order spatial differences is the best general method for linear hyperbolic equations available in the MOL1D package.

TABLE V.2. RESULTS ON THE MOL SOLUTION OF EQUATION (5.1) AT TIME $t = 2\pi$.
FOURTH ORDER SYMMETRIC DIFFERENCES WERE USED ON EVENLY SPACED MESH.

	CPU TIME (IN SECONDS)							
Mesh Points =	8		16		32		64	
Temporal Error =	10^{-4}	10^{-6}	10^{-4}	10^{-6}	10^{-4}	10^{-6}	10^{-4}	10^{-6}
Adams-Moulton								
Function iteration	0.07	0.13	0.11	0.18	0.23	0.40	0.78	1.69
Diagonal approx.	0.09	0.15	0.16	0.47	0.90	0.82	3.80	4.36
Tridiagonal approx.	0.11	0.18	0.16	0.27	0.40	0.52	0.98	1.47
Five Diagonal approx	0.12	0.20	0.18	0.32	0.46	0.62	1.11	1.62
Gear								
Function iteration	0.10	0.16	0.14	0.24	0.24	0.42	0.98	1.35
Diagonal approx.	0.12	0.21	0.24	0.29	0.83	0.92	3.73	4.39
Tridiagonal approx.	0.14	0.23	0.22	0.36	0.39	0.65	1.30	1.84
Five diagonal approx	0.17	0.25	0.25	0.41	0.44	0.73	1.36	1.97
	PHASE ERROR							
Adams-Moulton								
Function iteration	-7.4E-2	-7.4E-2	-4.4E-3	-4.9E-3	-3.8E-5	-3.0E-4	-7.0E-4	-1.7E-5
Diagonal approx.	-7.4E-2	-7.2E-2	-5.4E-3	-4.9E-3	-6.1E-4	-9.8E-4	-7.1E-5	-1.2E-5
Tridiagonal approx.	-7.4E-2	-7.4E-2	-4.7E-3	-4.9E-3	2.0E-4	-3.0E-4	1.5E-4	-1.9E-5
Five diagonal approx	-7.4E-2	-7.4E-2	-4.6E-3	-4.9E-3	2.1E-4	-3.1E-4	1.9E-4	-1.5E-5
Gear								
Function iteration	-7.3E-2	-7.4E-2	-3.4E-3	-4.9E-3	1.2E-3	-3.0E-4	3.2E-5	-7.4E-6
Diagonal approx.	-7.3E-2	-7.4E-2	-4.4E-3	-4.9E-3	-5.2E-4	-3.0E-4	-1.6E-3	-1.5E-5
Tridiagonal approx.	-7.3E-2	-7.4E-2	-3.4E-3	-4.9E-3	1.1E-3	-3.0E-4	2.4E-4	-1.4E-5
Five diagonal approx	-7.3E-2	-7.4E-2	-3.4E-3	-4.9E-3	1.1E-3	-3.0E-4	2.8E-4	-1.5E-5

TABLE V.3. RESULTS USING THE ADAMS-MOULTON METHOD WITH FUNCTION ITERATION FOR THE SOLUTION OF EQUATION (5.1) AT $t = 2\pi$.

Difference Scheme	Spatial Order	Mesh Points	CPU Time	Time Steps	PDE Calls	Avg. Temporal Order	Phase Error	Corrected L_2 Error
FFT		8	0.25	53	89	4.3	1.1E-5	4.3E-6
symmetric	4	8	0.13	53	89	4.3	-7.4E-2	3.7E-6
symmetric	2	8	0.12	50	83	4.2	-6.2E-1	2.4E-6
unsymmetric	2	8	0.27	56	98	4.3	1.0E+0	3.5E-1
unsymmetric	3	8	0.24	52	85	4.2	-7.4E-3	1.4E-1
FFT		16	0.47	58	122	4.3	6.1E-6	8.3E-6
symmetric	4	16	0.18	51	92	4.2	-4.9E-3	6.0E-6
symmetric	2	16	0.18	50	91	4.2	-1.6E-1	5.8E-6
unsymmetric	2	16	0.54	74	134	4.2	3.1E-1	6.3E-2
unsymmetric	3	16	0.37	51	92	4.2	-4.9E-3	2.2E-2
FFT		32	2.14	119	282	3.1	1.8E-6	1.2E-4
symmetric	4	32	0.40	61	116	4.2	-3.0E-4	6.5E-6
symmetric	2	32	0.31	51	92	4.2	-4.0E-2	5.7E-6
unsymmetric	2	32	1.74	142	273	3.3	8.0E-2	8.3E-3
unsymmetric	3	32	0.87	70	129	4.1	-3.0E-4	2.7E-3
FFT		64	6.72	252	512	3.0	1.1E-6	1.6E-4
symmetric	4	64	1.69	139	278	4.7	-1.7E-5	6.2E-6
symmetric	2	64	0.89	83	159	4.0	-1.0E-2	4.0E-6
unsymmetric	2	64	6.42	312	558	2.8	2.0E-2	1.0E-3
unsymmetric	3	64	4.01	164	322	4.8	-1.7E-5	3.5E-4

NEW YORK UNIVERSITY
COURANT INSTITUTE-LIBRARY
New York N.Y. 10012

Example 5.1b is the unidirectional wave equation

$$u_t = u_x , \qquad -\pi \leq x \leq \pi$$

(5.2)

$$u(x,0) = \sin(kx), \quad k = 1,2,\ldots,8$$

with periodic boundaries and the solution

$$u(x,t) = \sin k(x+t).$$

The table below shows when the different spatial approximations to (5.2) break down as the number of mesh points per wave length decreases from 32 to 4.

Phase Error

k	Points per Wave length	Unsymmetric Differences		Symmetric Differences		FFT
		2nd order	3rd order	2nd order	4th order	
1	32	8.0E-2	-3.0E-4	-4.0E-2	-3.0E-4	3.2E-6
2	16	6.1E-1	-9.8E-3	-3.2E-1	-9.8E-3	1.0E-5
3	10.7	1.9	-7.3E-2	-1.1	-7.3E-2	1.1E-5
4	8	-	-3.0E-1	-	-3.0E-1	2.5E-5
5	7.4	-	-8.7E-1	-	-8.7E-1	4.8E-5
6	5.3	-	-	-	-	5.1E-5
7	4.6	-	-	-	-	8.4E-5
8	4	-	-	-	-	1.6E-4

These results agree with the expected phase error given by formulas (1.6) and (1.7) for the symmetric finite differences. The Fourier method will approximate the solution accurately until there are only two points per wave length and therefore does very well in this test.

Example 5.2 is the simple diffusion equation

$$u_t = u_{xx}, \qquad -\pi \leq x \leq \pi \tag{5.3}$$
$$u(x,0) = \sin x$$

with periodic boundary conditions. The exact solution to (5.3) is

$$u(x,t) = e^{-t} \sin x .$$

Table V.5 lists the L_2 error of numerical solution and the computing times for a variety of integration methods all using fourth order spatial differencing. The table shows the computing times to be very dependent upon the stiffness of the differential difference equations and the accuracy of the Jacobian approximations.

As shown in Section 2, the equations for parabolic equations are very stiff for large N and accurate approximations of the Jacobian are necessary to prevent small time steps. This is evident in the large CPU times on the 128 point mesh by the function iteration and diagonal Jacobian approximation methods. These tests indicate that either Adams-Moulton or Gear's methods adequately treat diffusion equations if a banded approximation to the Jacobian is used. Further results using a tridiagonal Jacobian approximation and the Adams-Moulton method are in Table V.6.

TABLE V.5. RESULTS OF THE MOL SOLUTION OF EQUATION (5.3) AT t = 1.0 USING FOURTH ORDER SYMMETRIC DIFFERENCES.

	CPU TIME							
Mesh Points =	16		32		64		128	
Temporal Error =	10^{-4}	10^{-6}	10^{-4}	10^{-6}	10^{-4}	10^{-6}	10^{-4}	10^{-6}
Adams Moulton								
Function iteration	0.07	0.08	0.65	0.93	4.73	7.78	32.56	41.43
Diagonal approx.	0.06	0.08	0.50	1.43	5.34	5.25	15.01	39.49
Tridiagonal approx.	0.11	0.14	0.23	0.39	6.46	5.89	16.39	14.13
Five diagonal approx	0.14	0.15	0.38	0.48	0.23	1.08	15.55	16.57
Gear								
Function iteration	0.07	0.09	0.73	0.90	4.79	6.62	34.61	43.47
Diagonal approx.	0.08	0.22	1.17	1.47	4.86	6.54	6.72	73.83
Tridiagonal approx.	0.16	0.16	0.30	0.37	6.62	7.24	16.76	15.52
Five diagonal approx	0.19	0.18	0.42	0.43	0.28	1.20	15.94	13.45
	L_2 ERROR							
Adams Moulton								
Function iteration	7.6E-5	6.8E-5	6.0E-5	3.9E-6	2.3E-4	3.9E-8	1.0E-4	3.6E-5
Diagonal approx.	8.5E-5	6.9E-5	5.6E-4	7.9E-6	1.7E-4	7.9E-6	2.5E-4	4.7E-6
Tridiagonal approx.	3.9E-5	7.7E-5	1.6E-4	5.1E-6	3.0E-4	4.8E-7	2.0E-4	7.4E-5
Five Diagonal approx	4.9E-5	6.9E-5	2.3E-5	5.0E-6	6.3E-5	1.2E-6	1.6E-4	5.7E-7
Gear								
Function iteration	1.5E-4	6.7E-5	5.7E-4	1.2E-6	2.4E-4	1.7E-7	1.0E-4	5.0E-5
Diagonal approx.	7.9E-5	6.8E-5	2.8E-4	4.2E-6	8.4E-4	6.9E-6	6.5E-3	2.7E-5
Tridiagonal Approx.	5.2E-4	6.7E-5	9.9E-5	6.0E-6	3.3E-4	1.0E-6	4.6E-4	7.2E-7
Five diagonal approx	5.3E-5	6.7E-5	5.2E-5	5.2E-6	1.7E-4	1.0E-6	3.7E-4	1.8E-6

TABLE V.6. RESULTS USING THE ADAMS-MOULTON METHOD WITH A TRIDIAGONAL APPROXIMATION OF THE JACOBIAN FOR THE MOL APPROXIMATION TO EQUATION (5.3) AT t = 1.0.

Difference Scheme	Spatial Order	Mesh Points	CPU Time	Time Steps	PDE Calls	Avg. Temporal Order	Maximum Error	L_2 Error
FFT		8	0.10	21	35	3.0	1.4E-7	1.0E-7
symmetric	2	8	0.07	20	46	2.9	1.9E-2	1.3E-2
symmetric	4	8	0.07	21	47	3.0	1.5E-3	1.0E-3
FFT		16	0.44	61	107	2.4	2.1E-6	1.5E-6
symmetric	2	16	0.12	21	52	2.7	4.7E-3	3.3E-3
symmetric	4	16	0.14	22	55	2.7	9.7E-5	6.9E-5
FFT		32	4.46	449	631	2.0	8.7E-7	6.1E-7
symmetric	2	32	0.31	31	80	2.7	1.2E-3	8.4E-4
symmetric	4	32	0.39	38	93	2.7	7.2E-6	5.1E-6
FFT		64	29.32	1324	2170	1.9	1.2E-5	8.0E-6
symmetric	2	64	4.22	228	656	2.0	2.9E-4	2.1E-4
symmetric	4	64	5.89	292	888	2.0	1.3E-6	4.7E-7
symmetric	2	128	17.97	452	1485	2.0	7.3E-5	5.1E-5
symmetric	4	128	14.13	360	1093	2.0	2.9E-6	7.4E-7

In Example 5.3 we present results of the six initial-boundary value problems described in Table V.1. The purpose of these examples is to illustrate the stability and accuracy of the boundary approximation methods described in Section 1. All the problems used fourth order spatial differences and the Adams-Moulton method with a banded approximation to the Jacobian. The errors in Problems (5.3a-f) at $t = 2\pi$ are listed in the table below.

TABLE V.7

Example	16 Points		32 Points		64 Points	
	L_2 Error	Phase Error	L_2 Error	Phase Error	L_2 Error	Phase Error
5.3a	1.5E-2	-	3.2E-3	-	9.2E-4	-
5.3b	1.7E-2	-6.4E-3	3.3E-3	-4.3E-4	9.4E-4	-4.0E-5
5.3c	1.6E-2	-6.4E-3	3.2E-3	-4.3E-4	7.6E-4	-3.3E-5
5.3d	2.1E-3	-4.9E-3	1.6E-4	-3.6E-4	1.3E-5	-3.3E-5
5.3e	6.4E-3	2.3E-5	2.2E-3	-8.8E-4	8.6E-4	-3.9E-4
5.3f	8.5E-2	-	8.6E-3	-	9.4E-4	-

In Example 5.3a a plot of the numerical solution (not given here) as a function of x at time $t = 2\pi$ shows that most of the errors are in a neighborhood of $x = -\pi$. This indicates that they are due to inexact boundary approximations and that the error in Table V.7 is a measure of reflections caused by the differencing technique. In Example 5.3b the distortion of an incoming wave caused by inexact boundary approximations

is measured after a full cycle of the wave has entered from the right boundary. Example 5.3c measures the same type of distortion, the difference being that the boundary condition is posed as an ODE. The boundary conditions in Example 5.3d contain spatial derivatives and the procedure described in Section 1 was used to define the fictitious points needed in the boundary approximation. Table V.7 shows that the solution is well approximated using this method. In Example 5.3e the boundary conditions switch during the integration of the problem. Table V.7 indicates that there are no problems in solving equations with varying types of boundary conditions. This example was also solved by J. Gary in [9] using a third order boundary approximation and fourth order Runge-Kutta in time.

Example 5.3f computes approximations to an initial boundary value problem that is not in characteristic form using a boundary approximation method due to J. Gary [9]. Consider the system

$$\begin{pmatrix} u \\ v \end{pmatrix}_t = \begin{pmatrix} 3 & -4 \\ 2 & -3 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}_x, \quad 0 \leq x \leq 2\pi \tag{5.4}$$

with the boundary conditions

$$u(0, t) - 2v(0, t) = \sin t$$

$$u(2\pi, t) - v(2\pi, t) = \sin t$$

Initial conditions are chosen consistent with the solution

$$
\begin{aligned}
u(x,t) &= 2\sin(x+t) + \sin(x-t) \\
v(x,t) &= \sin(x+t) + \sin(x-t) \ .
\end{aligned}
\tag{5.5}
$$

The boundary conditions are proper since the system can be transformed by

$$
\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} 1 & -1 \\ -1 & 2 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}
$$

into

$$
\begin{pmatrix} w_1 \\ w_2 \end{pmatrix}_t = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}_x
$$

with the boundary conditions

$$
\begin{aligned}
w_1(2\pi,t) &= \sin t \\
w_2(0,\ t) &= -\sin t \ .
\end{aligned}
$$

We now incorporate the boundary conditions into the spatial difference operator for equation (5.4) without transforming the system into characteristic form. This is done by differentiating the boundary conditions with respect to t and combining with (5.4) to get the following differential boundary conditions

$$
\begin{aligned}
u_t &= 4u_x - 6v_x + \cos t \ , \qquad x = 0 \\
v_t &= \tfrac{3}{2}u_x - 2v_x - \tfrac{1}{2}\cos t
\end{aligned}
\tag{5.6a}
$$

and

$$
\begin{aligned}
u_t &= 2u_x - 3v_x + \cos t \ , \qquad x = 2\pi \\
v_t &= 3u_x - 4v_x - \cos t.
\end{aligned}
\tag{5.6b}
$$

Equation (5.4) was solved using equations (5.6) as boundary conditions and the error in u(x,t) is given in Table V.7. We do not recommend this procedure in general since there are no theoretical results to back the applicability of this method. We do offer it as an alternative method if standard techniques fail.

Example 5.4 illustrates the advantage of using a coordinate transformation to improve the accuracy in numerical solutions of the equation

$$
\begin{aligned}
u_t &= -u_x + \nu u_{xx}\,, && \nu = 0.003,\\
u(x,0) &= 2 + \cos \pi(2x-1), && 0 \le x \le 1
\end{aligned}
\tag{5.7}
$$

with the boundary conditions

$$u(0,t) = u(1,t) = 1\ .$$

The decay of the solution due to the diffusive term is small and the solution behaves like the solution of the associated hyperbolic problem (i.e. $\nu = 0$) for $x < 1$. Thus at $t = 1/2$ the solution to (5.7) is approximated by

$$u(x,t) \doteq \begin{cases} 2 + \cos 2\pi(x-1)\,, & \frac{1}{2} \le x < 1 \\ 1\,, & 0 \le x < \frac{1}{2} \end{cases}$$

with a sharp boundary layer in a small neighborhood of $x = 1$.

The unit interval was discretized into 64 evenly spaced mesh points and equation (5.7) was solved using fourth order spatial differences by MOL1D. The graph below shows the solution at $t = 0$ and $t = 1/2$ and demonstrates how reflections destroy the solution near the right boundary. The accuracy of the solution can be improved by a coordinate transformation of the form $Z = \sqrt{x}$. The transformed equation is

$$u_t = 2(\nu - z)u_z + 4\nu z^2 u_{zz}\ . \tag{5.8}$$

If we discretize evely in z space, more points are crowded near the boundary in the original x space. Figure 5.2b shows how the reflections disappear in the transformed equation. Similar results were obtained by Browning, Kreiss and Oliger [2] for this equation by refining the mesh in the original coordinate system.

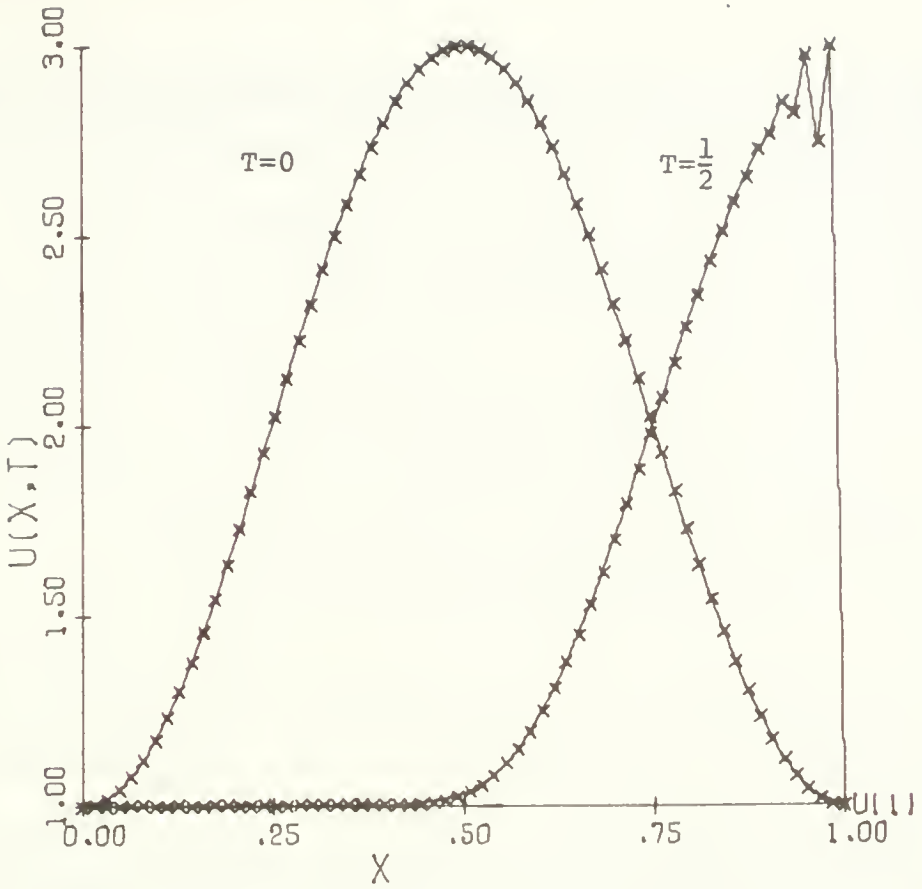


Figure 5.1a. MOL1D Solution of equation (5.7).

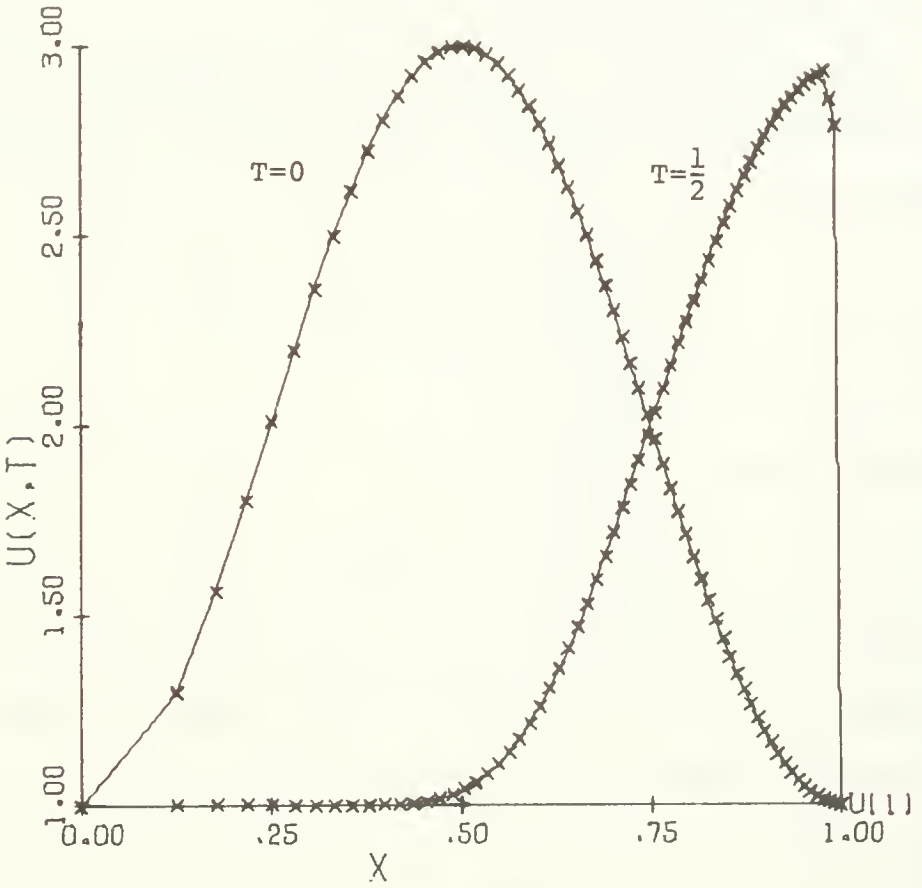


Figure 5.1b. MOL1D Solution of equation (5.8).

Example 5.5 Troesch's equation is the two-point boundary value problem

$$y_{xx} - n \sinh ny = 0 \tag{5.9}$$

where

$$y(0) = 0 , \quad y(1) = 1 .$$

The exact solution of (5.9) is given in [34] as

$$y(x) = \frac{2}{n} \sinh^{-1}[a \, sc(nt|1-a^2)] ; \tag{5.10}$$

$\sinh^{-1}$ is the inverse of the hyperbolic sine and sc is the Jacobian elliptic function with parameter $1-a^2$; $a = \frac{1}{2} y_x(0)$. This equation has become a standard test problem in solving two-point boundary value problems because of the difficulties encountered at the sharp boundary layer in the solution near $x = 1$.

To fit equation (5.14) into the framework of MOL, we view the solution as the stable equilibrium of the time dependent PDE

$$\begin{aligned} u_t &= u_{xx} - n \sinh nu \\ u(x,0) &= x \end{aligned} \tag{5.11}$$

with the boundary conditions

$$u(0,t) = 0 , \quad u(1,t) = 1 .$$

The equation was solved with $n = 10$ and $a = 0.0001791688853$ using Gear's method in time and fourth order centered differences in space on a 64 point uniform mesh.

Figure 5.2 is the graph of the PDE solution at various times. The steady state solution is reached at approximately t = 0.05.

It is clear from Figure 5.2 that the solution varies rapidly in the region near x = 1. Therefore, the coordinate transformation method is indicated. The transformation $z = x^{1/m}$, $m \geq 1$, stretches the boundary layer region in a neighborhood of x = 1. The transformed equation is

$$(5.12) \quad u_t = (mz^{m-1})^2 u_{zz} + m(m-1)z^{m-2}u_z - n \sinh nu$$

with boundary conditions

$$u(0,t) = 0 , \qquad u(1,t) = 1 .$$

Equation (5.12) reduces to (5.11) for m = 1 as it should. The equation was solved for n = 10 and m = 1,2,...,30 and in each case the steady state was determined. The table below lists the maximum deviation of these steady state solutions from the exact solution (5.10).

m	Maximum Error
1	0.055
2	0.035
3	0.025
4	0.019
5	0.015
7	0.009
10	0.005
13	0.003
15	0.005
20	0.013
25	0.022
30	0.031

It is clear from the above table that the coordinate transformation increases the accuracy of the numerical solution significantly for $10 \leq m \leq 15$.

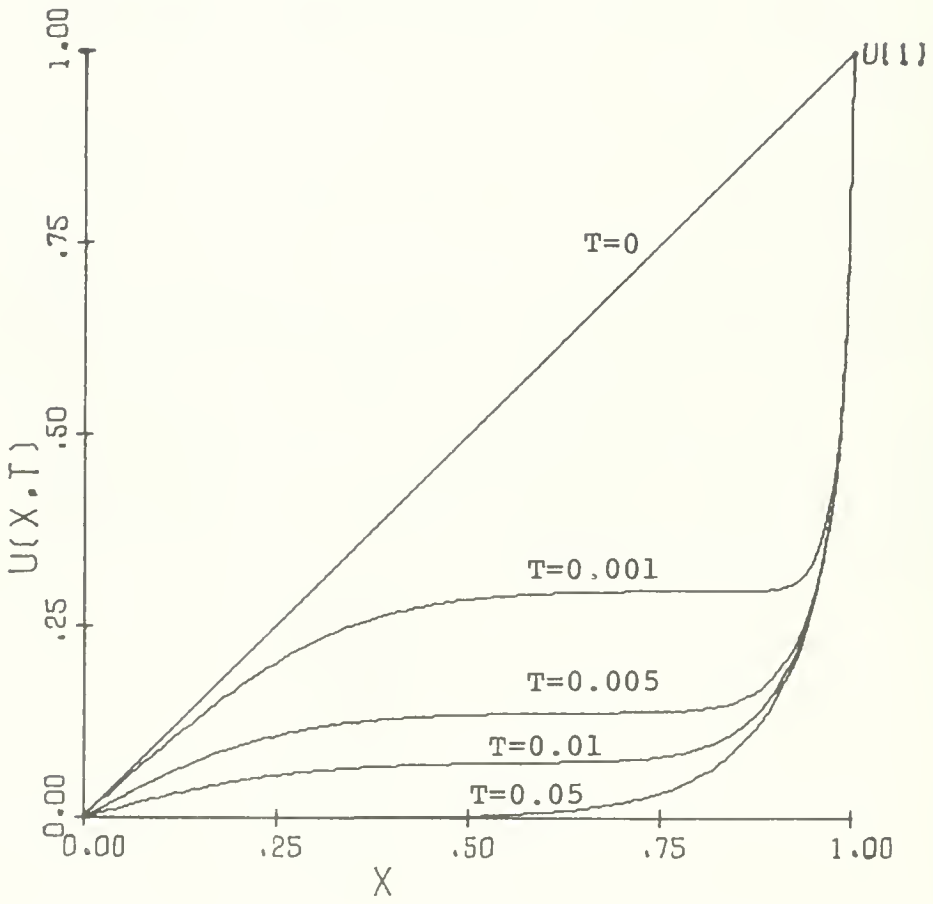


Figure 5.2. MOL1D solution of equation (5.11) with n = 10.

Example 5.6 Burger's equation

$$u_t = -\frac{1}{2}(u^2)_x + \nu u_{xx}, \quad \nu = 0.003, \quad 0 \le x \le 1, \tag{5.13}$$

is solved for initial data and boundary conditions consistent with the exact solution

$$u(x,t) = \frac{1}{10}\frac{e^{-a} + 5e^{-b} + 10\, e^{-c}}{e^{-a} + e^{-b} + e^{-c}}, \tag{5.14}$$

where

$$a = \frac{50}{3}\left(x - \frac{1}{2} + \frac{99}{20}t\right)$$

$$b = \frac{250}{3}\left(x - \frac{1}{2} + \frac{3}{4}t\right)$$

$$c = \frac{500}{3}\left(x - \frac{3}{8}t\right)$$

Burger's equation is typical of nonlinear dissipative equations that arise when artificial viscosity is added to stabilize approximations of nonlinear conservation laws. Figure 5.3 is a graph of the numerical solution at selected times and Table V.9 shows the accuracy of these calculations. The time variable was integrated very accurately (EPS = 10^{-5}) to isolate the errors due to the spatial discretization.

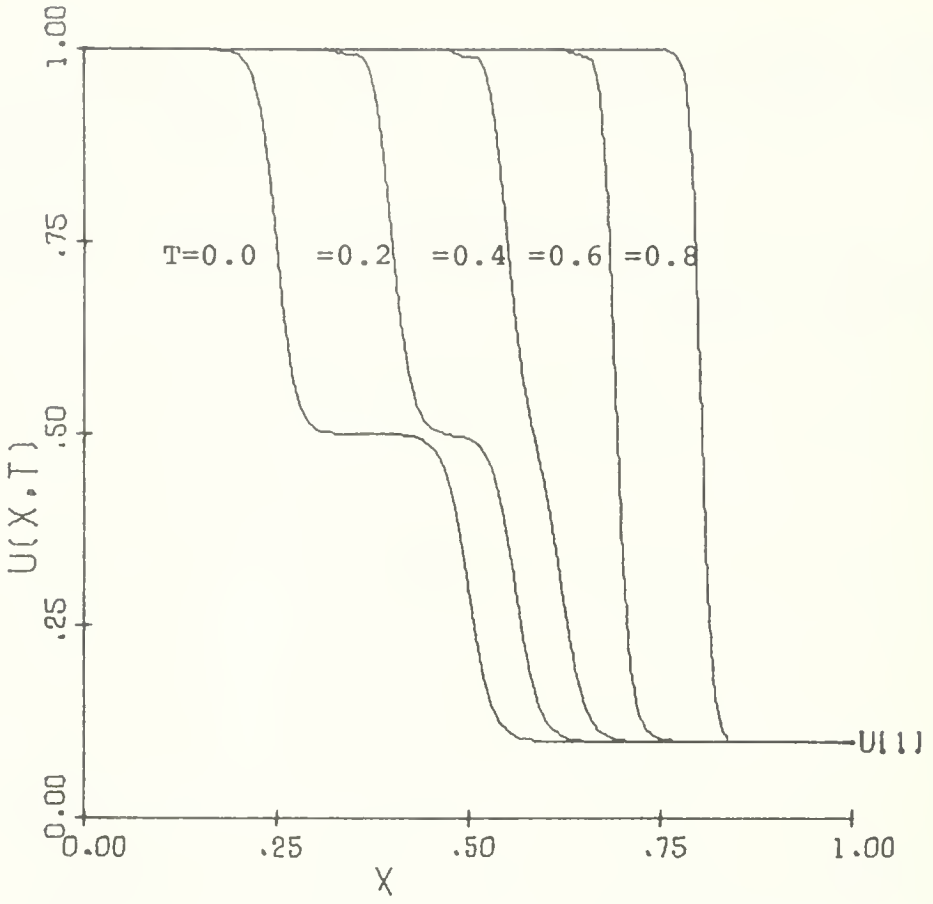


Figure 5.3. Solution to Burger's Equation (Example 5.6).

TABLE V.9. NUMERICAL ERRORS IN THE APPROXIMATION TO BURGER'S EQUATION (5.13) AT t = 1.0.

Space Difference	Spatial order	L_1 Error		L_2 Error		L_∞ Error	
		101 points	201 points	101 points	201 points	101 points	201 points
symmetric	4	7.9E-4	3.4E-5	2.6E-3	1.6E-4	1.5E-2	1.4E-3
symmetric	2	1.9E-3	4.1E-4	8.1E-3	1.8E-3	6.0E-2	1.7E-2
unsymmetric	3	5.1E-4	7.4E-5	1.8E-3	2.9E-4	1.1E-2	2.1E-3
unsymmetric	2	2.5E-3	7.5E-4	1.0E-2	3.2E-3	6.9E-2	2.6E-2

The CPU times for the unsymmetric difference were approximately 30% greater than the symmetric differences. This increase in computer time results from estimating the direction of the characteristics of the associated hyperbolic problem (i.e. $\nu = 0$) at each mesh point.

This problem was also solved by Madsen and Sincovec in [39] using the GEARB package in time and second order centered differences in space. Table V.9 agrees with their results for the same combinations of time and space methods.

Example 5.7 is the MOL solution of the Eulerian equations of gas dynamics. These equations can be written in the form [14]

$$W_t + F(W)_x = 0 \tag{5.15}$$

$$W = \begin{pmatrix} \rho \\ m \\ E \end{pmatrix}, \quad F(W) = uW + \begin{pmatrix} 0 \\ p \\ pu \end{pmatrix}$$

where

ρ = mass density

m = momentum

E = total energy per unit volume

p = pressure

$u = m/\rho$ = velocity

We assume that the equation of state is that of a polytropic gas, i.e.,

$$p = (\gamma - 1)(E - \tfrac{1}{2}\rho u^2)$$

where

γ = constant.

We solve (5.15) with the initial conditions

$$\begin{cases} \rho(x,0) = 1.0 \\ m(x,0) = 0.0 \\ E(x,0) = 1.5 \end{cases}, \quad x < \frac{3}{4}; \qquad \begin{cases} \rho(x,0) = 0.125 \\ m(x,0) = 0.0 \\ E(x,0) = 0.15 \end{cases}, \quad x \geq \frac{3}{4}$$

$$\gamma = \frac{5}{3}$$

and reflecting boundary conditions at x = 0 and x = 2 of the form

$$\rho_x = 0$$

$$m = 0$$

$$E_x = 0 \ .$$

Artificial dissipation is needed to stabilize the calculations and achieve proper entropy production. The actual equation solved was a fourth order spatial discretization of

$$W_t + F(W)_x = \nu W_{xx} \ .$$

ν is the coefficient of the artificial dissipation and must be chosen large enough to damp out instabilities at the shock interface and yet small enough not to smear the discontinuities; $\nu = 0.01$ was found to be sufficient for this problem after a few trial runs. Calculations with $\nu = 0.005$ are stable but ripples develop at the back of the shocks. Figures 5.4a-e show graphs of the solution using both $\nu = 0.01$ and $\nu = 0.005$ at selected times. In the calculations fourth order centered differences were used on a uniform mesh of 81 points. The solution consists initially of a rarefaction wave traveling to the left, a contact discontinuity traveling slowly to the right and a shock traveling much faster to the right. The shock reflects from the boundary in Figure 5.4d and collides with the contact discontinuity in Figure 5.4e. When using the smaller artificial dissipation the discontinuities are

sharper but post shock oscillations developed due to insignificant dissipation. If the mesh is refined or sixth order differences are used, then less dissipation is needed to stabilize the system and the oscillations disappear for $\nu = 0.005$.

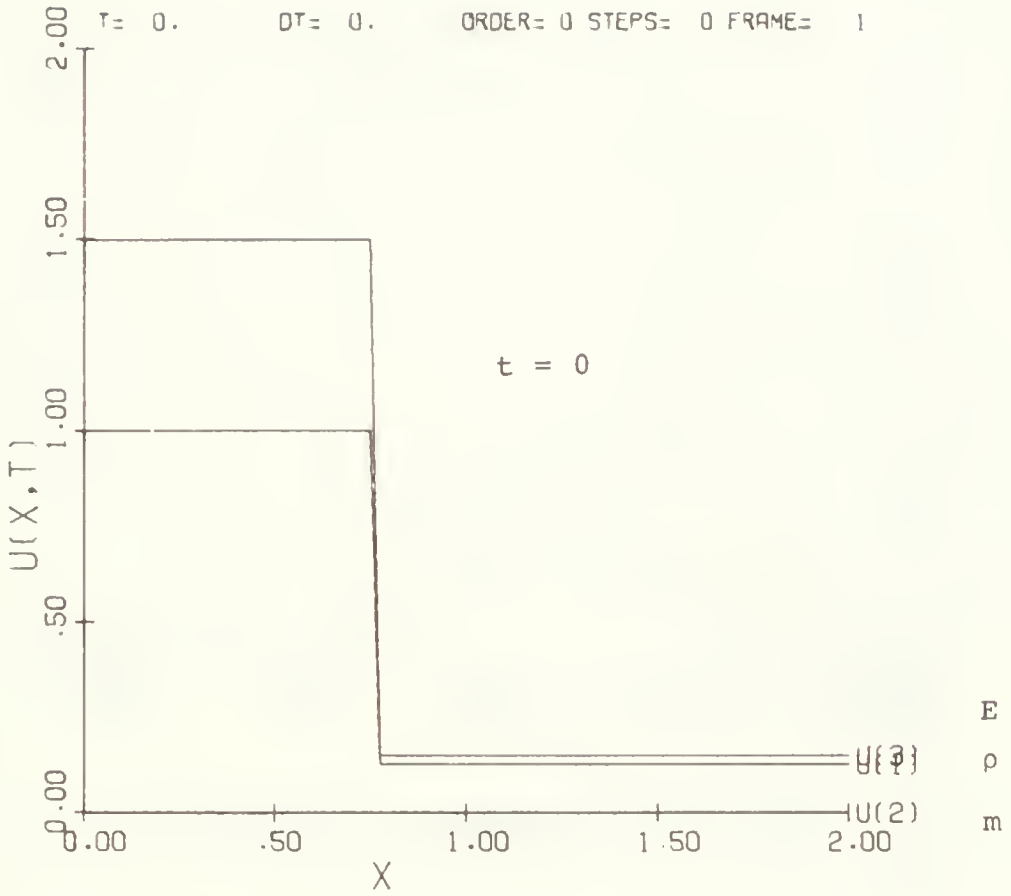
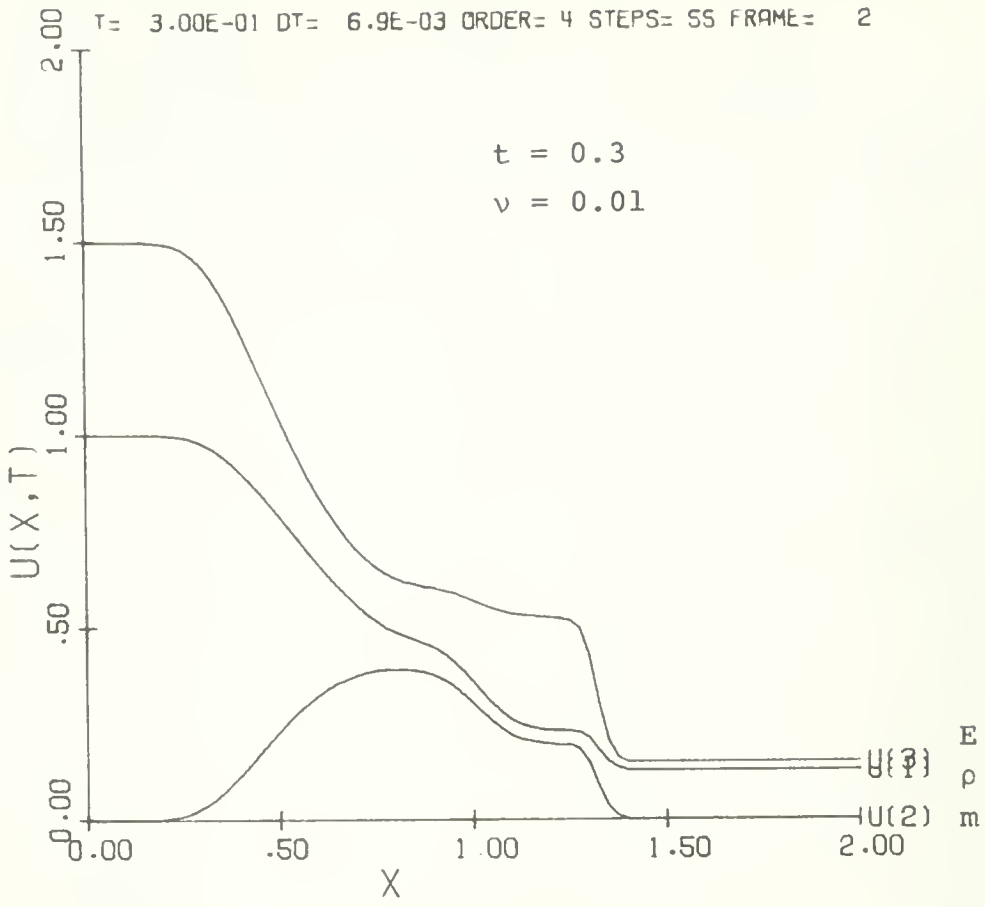


Figure 5.4a. MOL1D solution of the Eulerian equations of gas dynamics (5.15) with two different amounts of artificial viscosity. Fourth order symmetric differences were used on a uniformly spaced mesh of 81 points. $u_1 = \rho$, $u_2 = m$, $u_3 = E$.


T= 3.00E-01 DT= 6.9E-03 ORDER= 4 STEPS= 55 FRAME= 2
t = 0.3
ν = 0.01
U(X,T)
2.00
1.50
1.00
.50
0.00
0.00
.50
1.00
1.50
2.00
X
U(3)
U(1)
U(2)
E
ρ
m

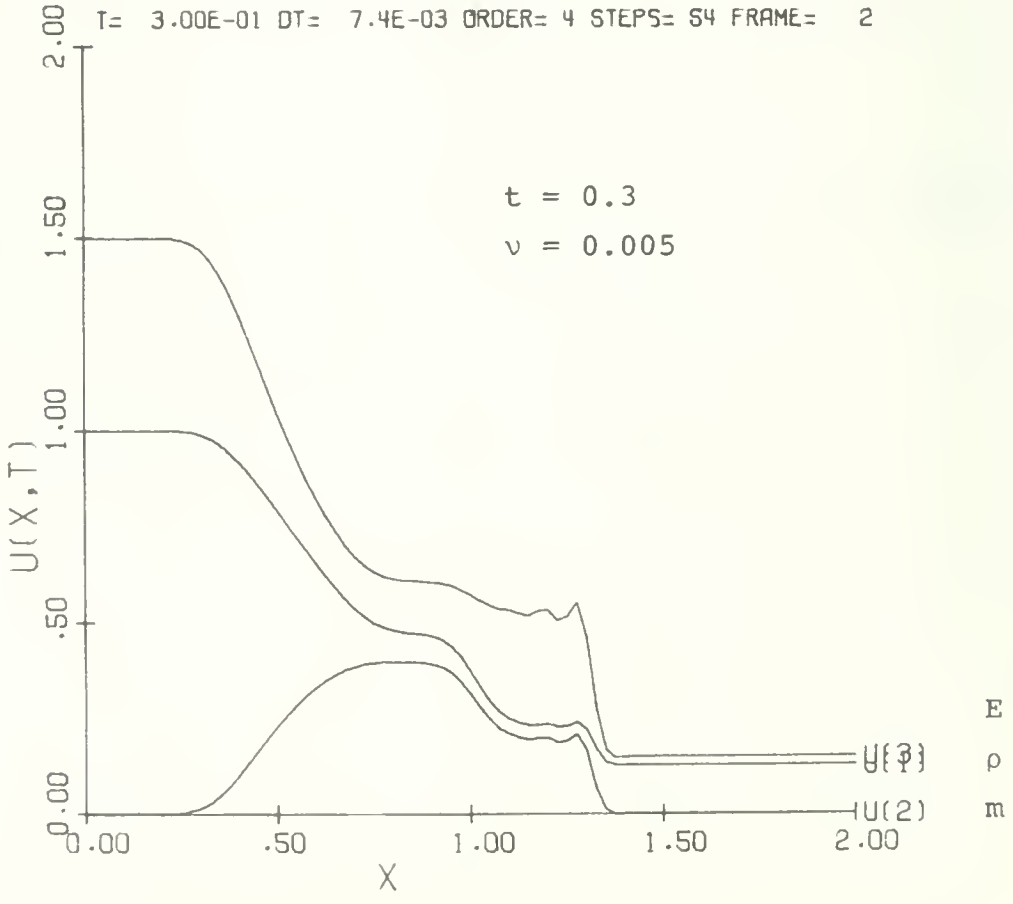
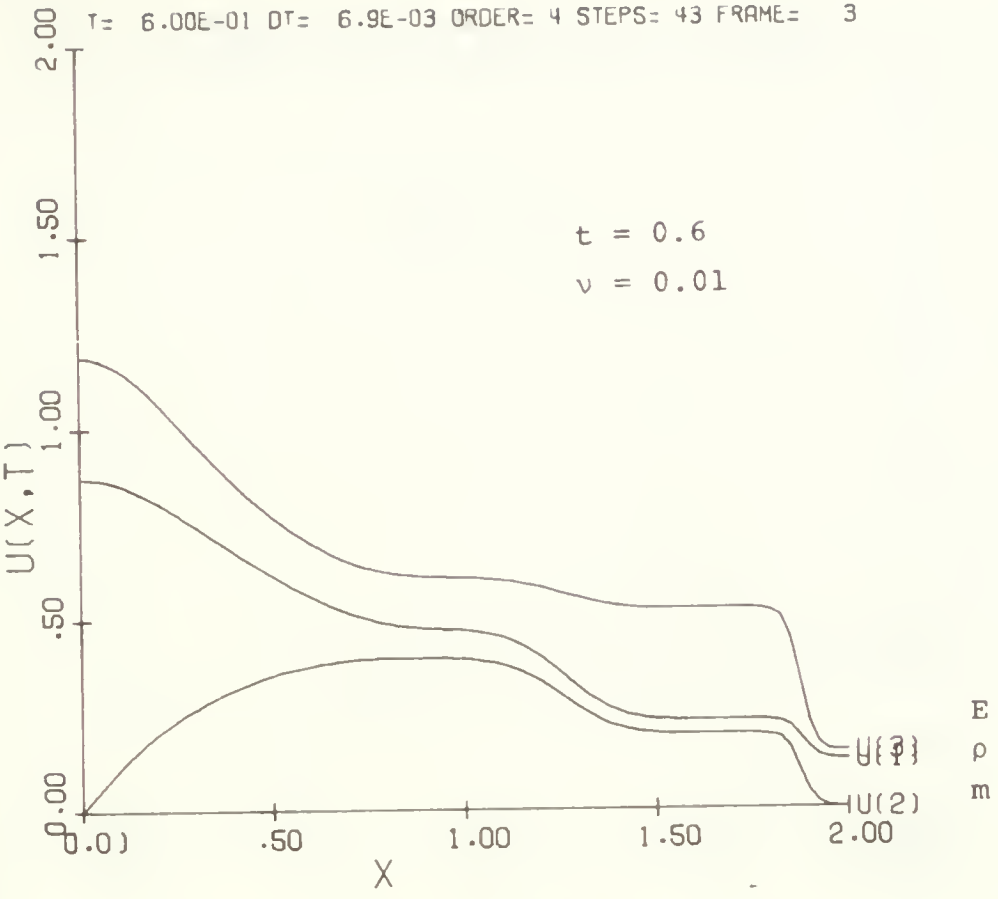

T= 3.00E-01 DT= 7.4E-03 ORDER= 4 STEPS= 54 FRAME= 2
t = 0.3
ν = 0.005
U(X,T)
2.00
1.50
1.00
.50
0.00
0.00
.50
1.00
1.50
2.00
X
U(3)
U(1)
U(2)
E
ρ
m

Figure 5.4b.


T= 6.00E-01 DT= 6.9E-03 ORDER= 4 STEPS= 43 FRAME= 3
t = 0.6
ν = 0.01
U(X,T)
2.00
1.50
1.00
.50
0.00
.0)
.50
1.00
1.50
2.00
X
U(3)
U(1)
U(2)
E
ρ
m

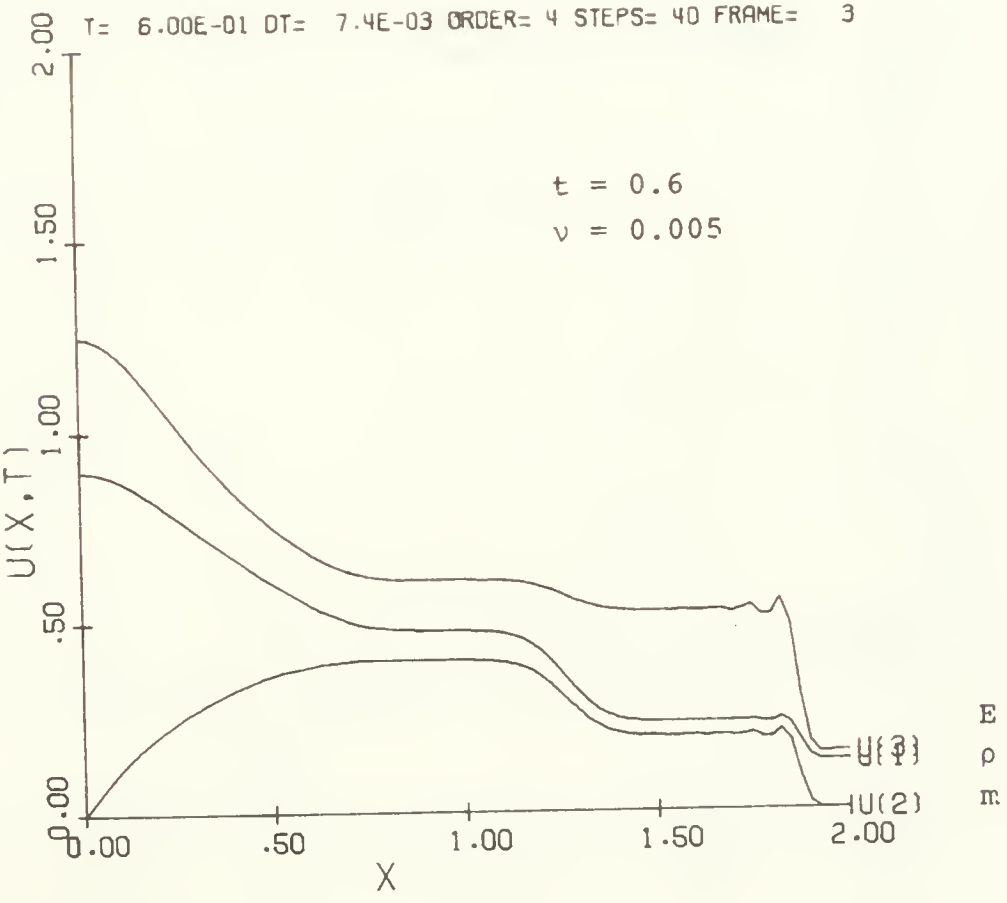
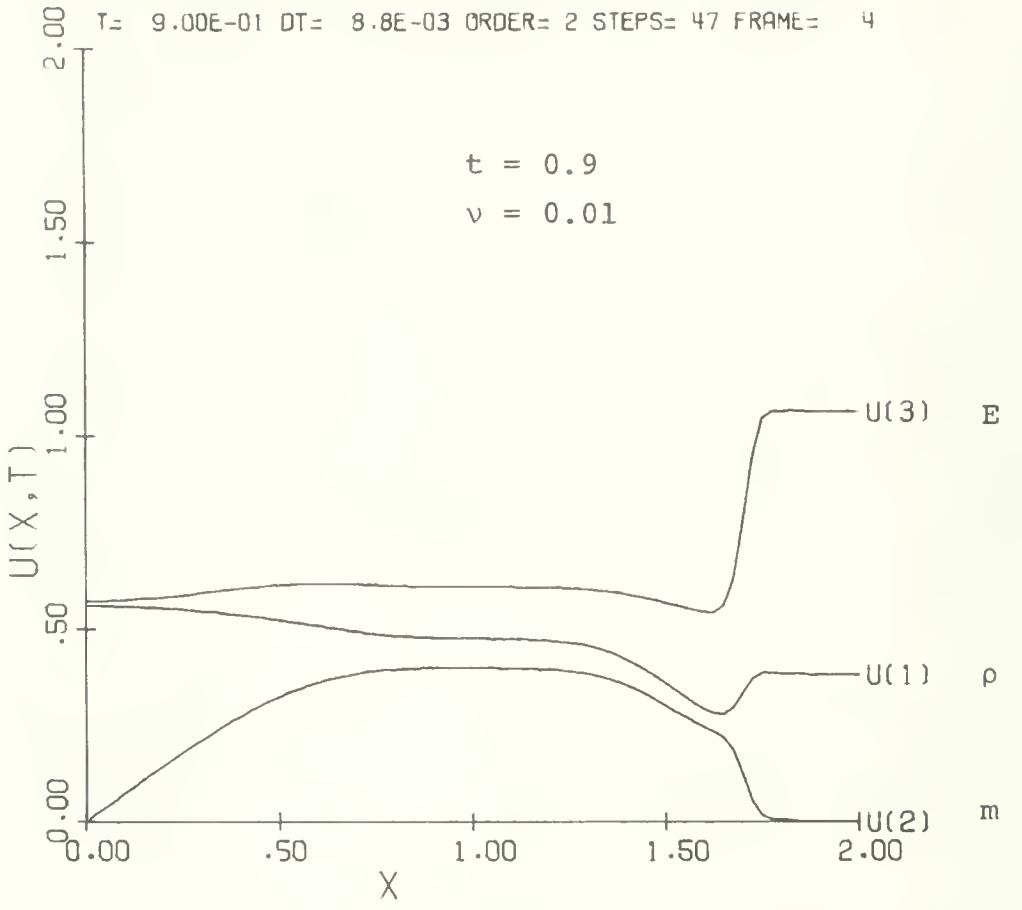

T= 6.00E-01 DT= 7.4E-03 ORDER= 4 STEPS= 40 FRAME= 3
t = 0.6
ν = 0.005
U(X,T)
2.00
1.50
1.00
.50
0.00
0.00
.50
1.00
1.50
2.00
X
U(3)
U(1)
U(2)
E
ρ
m

Figure 5.4c


T= 9.00E-01 DT= 8.8E-03 ORDER= 2 STEPS= 47 FRAME= 4
t = 0.9
ν = 0.01
U(3) E
U(1) ρ
U(2) m
U(X,T)
X
0.00
.50
1.00
1.50
2.00

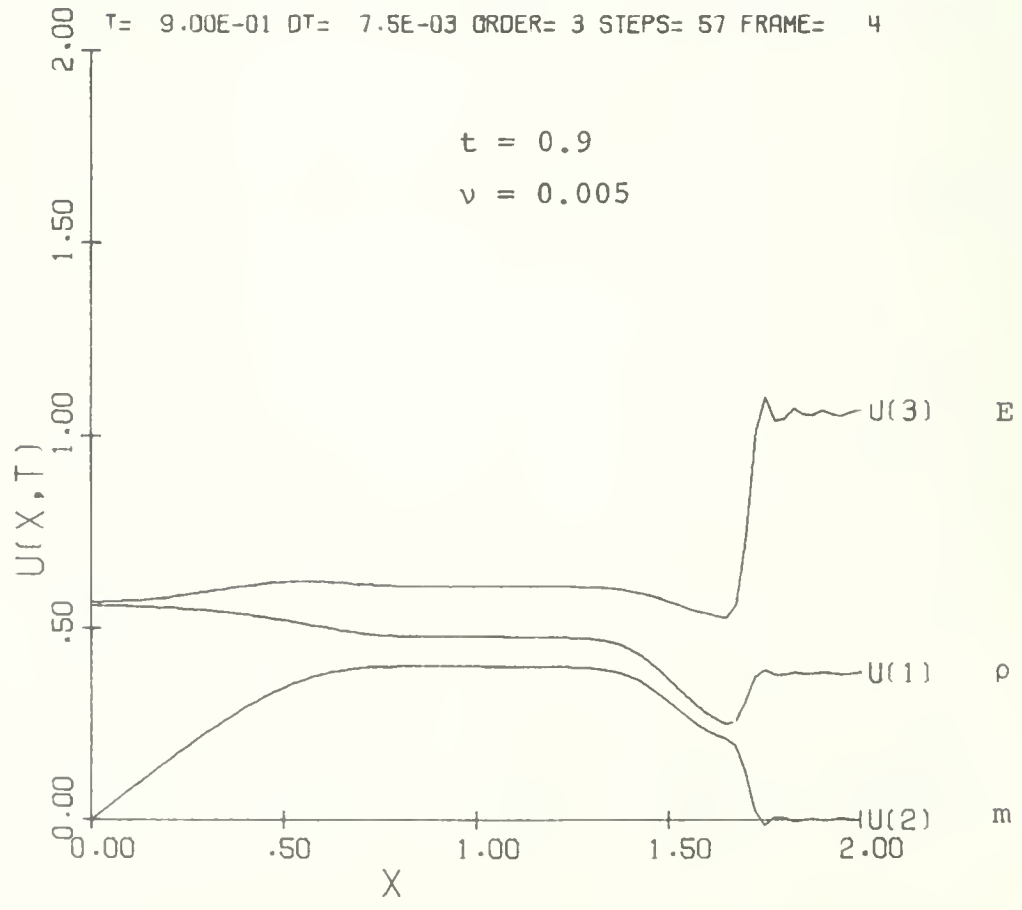
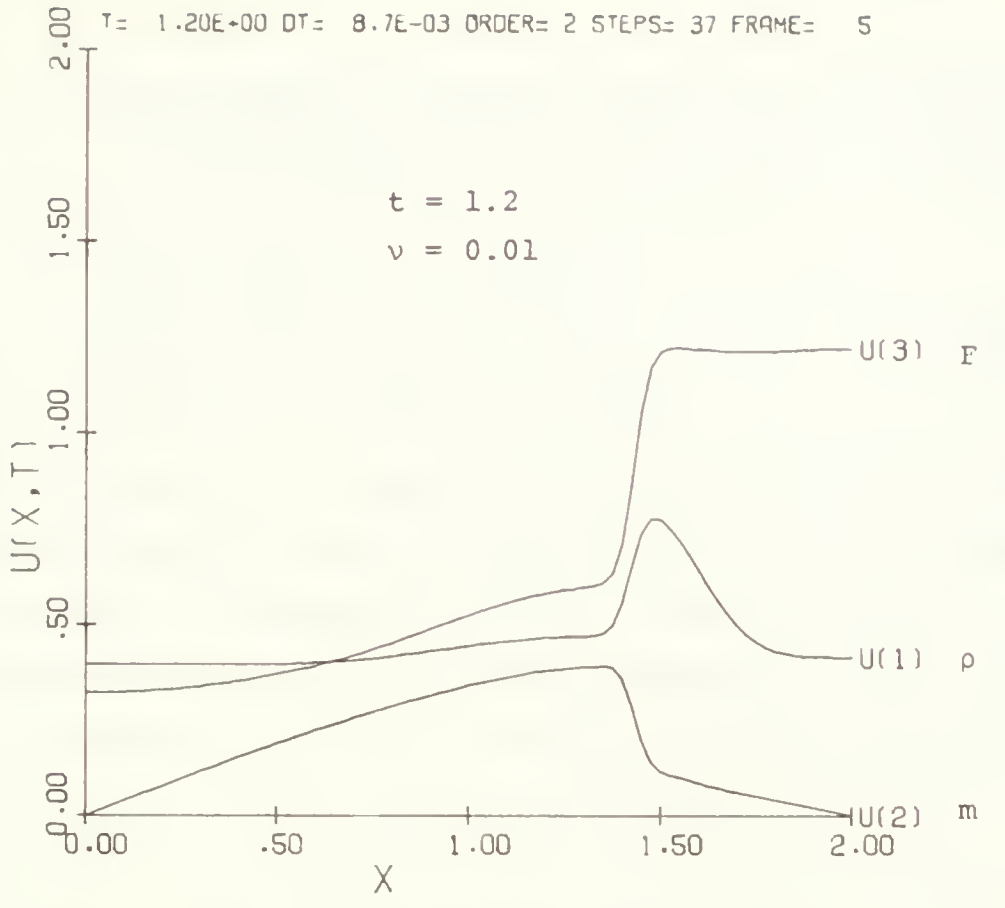

T= 9.00E-01 DT= 7.5E-03 ORDER= 3 STEPS= 57 FRAME= 4
t = 0.9
ν = 0.005
U(3) E
U(1) ρ
U(2) m
U(X,T)
X
0.00
.50
1.00
1.50
2.00

Figure 5.4d


T= 1.20E+00 DT= 8.7E-03 ORDER= 2 STEPS= 37 FRAME= 5
t = 1.2
ν = 0.01
U(3) F
U(1) ρ
U(2) m
U(X,T)
X
0.00
.50
1.00
1.50
2.00

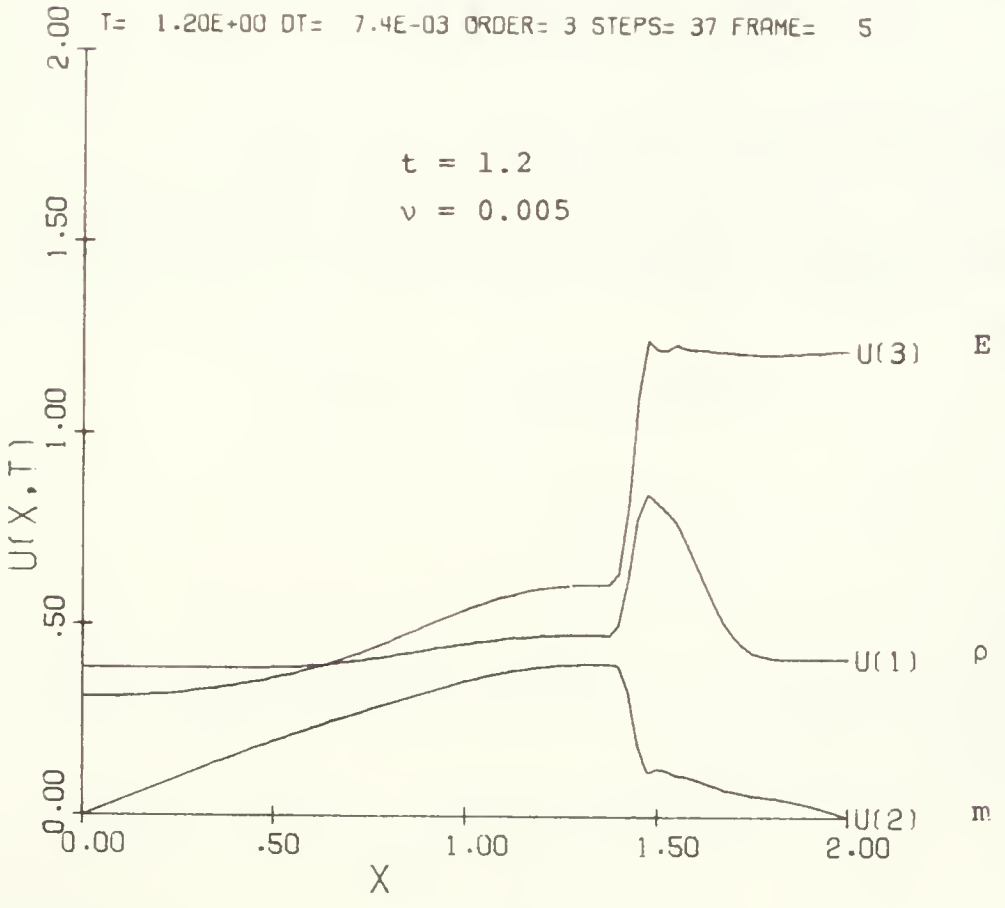

T= 1.20E+00 DT= 7.4E-03 ORDER= 3 STEPS= 37 FRAME= 5
t = 1.2
ν = 0.005
U(3) E
U(1) ρ
U(2) m
U(X,T)
X
0.00
.50
1.00
1.50
2.00

Figure 5.4e

Example 5.8 is based on a system of autocatalytic chemical reactions described by Robertson [35]:

$$
\begin{aligned}
A &\xrightarrow{k_1} B \quad , \qquad k_1 = 0.04 \\
B + C &\xrightarrow{k_2} A + C \quad , \qquad k_2 = 10^4 \\
2B &\xrightarrow{k_3} C + B \quad , \qquad k_3 = 3\times 10^7 .
\end{aligned}
\tag{5.16}
$$

The differential equations describing the kinetics of this system are very stiff and Robertson's model has become a standard test problem for stiff ODE methods. We modify the equations to include spatial variations and passive diffusion. The corresponding system of PDEs for these reactions is

$$
\begin{aligned}
u_t &= -k_1 u + k_2 vw + d u_{xx} \quad , \qquad 0 \le x \le 1 \\
v_t &= k_1 u - k_2 vw - k_3 v^2 + d v_{xx} \\
w_t &= k_3 v^2 + d w_{xx}
\end{aligned}
\tag{5.17}
$$

where the diffusion coefficient $d = 0.1$ and $u = [A]$, $v = [B]$ and $w = [C]$ denote the concentrations of the chemical components. We take initial conditions of the form

$$
u(x,0) = \begin{cases} 1.5 , & x \ge \frac{1}{2} \\ 0.5 , & x < \frac{1}{2} \end{cases}
$$

$$
v(x,0) = 0
$$

$$
w(x,0) = 0 \quad .
$$

At the boundaries the solution is constrained to satisfy the zero flux conditions

$$
u_x = v_x = w_x = 0 \quad , \qquad x = 0,1.
$$

As is typical in diffusion reaction problems the solution has a very quick initial transient state followed by a smooth transition to the steady state solution.

The difficulties in numerically solving chemical reaction diffusion equations are in the temporal integration and not the spatial approximation. Therefore we use fourth order differences on a very coarse grid of 8 mesh points to reduce the computer time for this example. Other numerical experiments of the author with MOL1D indicate that the behavior of the methods is the same for finer meshes.

The system (5.9.1) was integrated from $t = 0$ to $t = 10^3$. This final time is well into the smooth variation of the solution into steady state solution. The accuracy of the solutions was checked by the conservation of mass and the consistency of the numerical solutions computed by the different methods. The error of the solutions was approximately the same ($< 0.01\%$) for the different integration methods. However there are extreme differences in computer running times as is clear from the table below.

TABLE V.10. COMPUTER TIMES (IN SECONDS) FOR MOL1D SOLUTION OF (5.17)

MJAC	Jacobian Approximation	Adams-Moulton		Gear	
		EPS= 10^{-6}	10^{-8}	EPS= 10^{-6}	10^{-8}
1	Function iteration	*	*	*	*
2	Diagonal approximation	-	117	-	65
3	Banded approximation bandwith = 3	*	109	108	115
4	Banded approximation band width = 9	32	21	23	21
5	Banded approximation band width = 15	66	24	22	23

* unable to complete problem after two minutes of computer time.

- incorrectly integrated solution.

MJAC = Jacobian method flag used by MOL1D.

Notice that the computing time of the Adams-Moulton methods decreases when more stringent temporal accuracy restrictions are imposed. When the error tolerance (EPS) was set to 10^{-4} corrector convergence was not achieved by any method and an error message was returned by the GEARB package. The diagonal approximation methods decieved themselves when the error tolerance parameters was set to 10^{-6} and converged to the wrong solution. In this example the simple analysis of Section 3 for the optimal time-space discretization by balancing the time and space errors does not hold. The overall efficiency is improved by integrating the time variable much more accurately than the spatial derivatives are approximated. A similar occurrence was observed when integrating the highly stiff MOL approximation for the traveling wave solutions of the KdV equation ($u_t+uu_x+u_{xxx}= 0$).

6. DISCUSSION

Solving initial value problems for PDEs is a highly complicated process. In this paper we show by analysis and examples that MOL, although no panacea, is an effective approach for a broad class of problems, including some particularly troublesome ones. Using MOL we can successfully solve linear and nonlinear hyperbolic equations with or without discontinuities, linear and nonlinear parabolic equations, including those arising in reaction-diffusion process. Systems are easily handled as single equations, and a wide variety of boundary conditions can be accommodated including most that arise in applications.

The major advantages of MOL are that initial value problems can be solved very accurately with a minimum of programming effort and with moderate computer cost.

Some future modifications of the package include:

(a) The usefulness of unsymmetric differences will be critically evaluated and may be eliminated in future versions

(b) Unequally spaced meshes will be allowed as an option

(c) An attempt will be made to include a convenient way for the user to designate the storage needed to solve a particular problem.

(d) Subroutines UDER and FDER will be restructured to eliminate redundant coding in the derivative calculations and allow the user to easily use a spatial deriva-

tive approximation not explicitly provided by the package.

(e) It will be possible to print and plot variables other than the integration variables.

The package is available in the User's Library at the ERDA Mathematics and Computing Laboratory at the Courant Institute of New York University.

The author plans to distribute the package through the Argonne Code Center at Argonne National Laboratory by the Fall of 1977. Comments and suggests for improving the package are encouraged and should be mailed to

Dr. James M. Hyman
Group T-7, MS-233
Los Alamos Scientific Laboratory
Los Alamos, N. M. 87545

A complete user's manual describing the algorithms used by the package is being prepared and will be released simultaneously with the package. The manual will include examples to illustrate the performance of the code (successes as well as failures) over a wide range of problems. The manual will also include explanations of how to convert the code to other systems, including a detailed description of the graphics subroutine.

We are confident that after its shakedown cruise MOL1D will be a useful, usable and used item in the library of every scientific computing center.

APPENDIX. Calling Sequence and Structure of MOL1D

We now describe the arguments of subroutine MOL1D and the routine that the programmer needs to supply to use the package. The arguments of MOL1D describe how the system of NPDE partial differential equations are to be solved. The initial conditions for the PDE enter the routine in an array UZ defining the solution on a mesh XM of NPTS points. The user must describe the kind of boundary conditions (KBC) for the equation and has the option of describing the type of equation (KEQN) that is to be integrated (e.g. hyperbolic or parabolic, linear or nonlinear). The MOL approximation to be used is determined by the temporal integration method (METH), the error per time step (EPS) and the spatial method order (MORD). The user has the option of allowing the package to choose a time integration method based on the results of numerical testing as shown in Section 5. The equations are integrated from the initial time TINT to the final time TLAST with the output at user prescribed times in the array TOUT. The character of the output (MOUT) may be either printed or graphics. The printed output is written on tape 3 and the graphics use standard Calcomp plotting routines.

The complete calling sequence for MOL1D is

```
CALL MOL1D(NPDE,NPTS,KEQN,KBC,METH,EPS,MORD,TINT,TLAST,
          MOUT,TOUT,UZ,XM,KMOL) .
```

The arguments of MOL1D are defined as follows:

NPDE = number of partial differential equations

NPTS = number of spatial mesh points

KEQN = descriptor of the kind of equation that is to be solved. This variable is used to pick the initial time step and the ODE method if the default option is chosen. KEQN is greater than zero if the equation is linear and less than zero otherwise. It has the following values and meanings:

- ± 1 hyperbolic equation
- ± 2 parabolic equation
- ± 3 equation in conservation form and/or solution has discontinuities
- -4 highly nonlinear equation

KBC = descriptor of the kind of boundary conditions for the equation with the following values and meanings:

- 1 periodic boundary conditions
- 2 nonperiodic boundary conditions of the form $au+bu_x = c$, or some boundaries may be defined by an ODE, be free boundaries, or have no boundary conditions prescribed. If $b \neq 0$ then a,b, and c may be functions of t and u. Otherwise a and c may be functions of t only.
- 3 boundary conditions as in KBC=2 except that those defined by an ODE or PDE include spatial derivatives not calculated at the interior points.

METH = the ODE method flag used to set the calling arguments in GEARB. If either of the default options is used, METH returns equal to the method selected. This method is chosen by the values of KEQN, KBC, NPTS and NPDE. METH has two decimal digits, MODE and MJAC (METH = 10*MODE + MJAC). MODE is the ODE method indicator with the following values and meanings:

MODE= 0 MOL1D selects either Adam's-Moulton or Gear's method

MODE= 1 implicit Adams-Moulton methods

MODE= 2 Gear's stiffly stable methods:

MJAC is the corrector iteration method indicator, with the following values and meanings:

MJAC= 0 MOL1D selects one of the following methods:

MJAC= 1 Jacobian approximated by the identity matrix (function iteration)

MJAC= 2 Jacobian approximated by a diagonal matrix

MJAC= 3 Jacobian approximated by a banded matrix, band width = NPDE

MJAC= 4 Jacobian approximated by a banded matrix, band width = 3 NPDE

MJAC= 5 Jacobian approximated by a banded matrix, band width = 5 NPDE

EPS = time integrator local error tolerance parameter Estimates of the time step error at the spatial grid points for all the PDEs is kept less than EPS

in the root-mean-square (RMS) norm. EPS is also used to choose the initial time step.

MORD = a matrix dimensioned MORD(NPDE,3) indicating which derivatives of the PDE variables are needed and what type of approximation is to be used. MORD(I,1), MORD(I,2) and MORD(I,3) refer to $U(I)_x$, $U(I)_{xx}$ and $f_I(U)_x$ respectively. MORD has the following possible values and meanings:

6 sixth order centered differences
4 fourth order centered differences
2 second order centered differences
1 fast Fourier transform approximation
0 derivative does not appear in the equation
-1 fast Fourier transform with linear filtering of the higher modes.
-2 second order centered differences of unsymmetric second order differences oriented into the characteristics of $u_t = f_x$. The centered differences are used if the characteristics are perpendicular to the x axis.
-4 fourth order centered differences or unsymmetric third order differences oriented into the characteristics of $u_t = f_x$.

TINT = the initial value of the independent variable t

TLAST = the final integration time when control is returned to the calling program. TLAST must be greater than or equal to TINT. If the ODE integrator fails to converge during integration TLAST returns equal to the last integration time.

MOUT = an input parameter to indicate the character of the output. If MOUT is less than zero then the solution is printed at each time step for |MOUT| steps. TOUT(I) returns the time of the Ith time step. If MOUT is greater than or equal to zero, it has the following values and meanings:

0 No output is printed or plotted

1 The solution is printed at the preselected times in the array TOUT. The printing is in subroutine SOLPRT on Tape 3. This subroutine may be rewritten by the user to change the output format.

2 The solution and its time derivative are printed at the times in TOUT.

3 The solution is printed and plotted at the times in TOUT. The plotting is in subroutine SOLPLT. An alternative subroutine may be supplied by the user if he wishes to provide his own graphics routine.

4 The time derivative of the solution is printed and the solution is both printed and plotted at the times in TOUT.

5 The solution is plotted at the times in TOUT.

6 The solution is plotted rotated 90° on 16MM film to be viewed as a movie. At the Courant

Computer Center this option is available only if either the FILMF or MOVIEF library routines are loaded.

(See note on the graphics option following the calling sequence description.)

TOUT = an input array of the times that the solution is to be printed or plotted if MOUT > 0. The solution is interpolated at these times from its values at the actual integration times by Taylor series. If MOUT < 0 then the first |MOUT| printout times are returned in TOUT.

UZ = a matrix dimensioned UZ(NPDE, NPTS) of the dependent variable U . On input UZ(I,J) contains the value of the I-th PDE component at the J-th mesh point at time t = TINT. On output, UZ returns the value of the solution at time = TLAST.

XM = an array of length NPTS containing the uniformly spaced mesh points.

KMOL = an indicator flag used for both input and output. On input it has the following values and meanings:

K > 0 Call a debugging dump at every K-th printout time. In a dubugging dump the solution, all relevant spatial and temporal derivatives and error estimates are printed.

K = 0 No dumps are called unless an error is encountered.

K < 0 A dump is called on the first and last step

of the problem.

On output, KMOL returns

0 if the integration was successful

K < 0 The integration was halted after encountering a fatal error. A message is printed and K returns with the following values and meanings:

-1 integration was halted after failure to pass the error test even after the time step, DT, was reduced by a factor of 10^{10} from its original value.

-2 error test failure indicating EPS is too small for this problem.

-3 integration was halted after failing to achieve corrector convergence. Increase MJAC if possible for a more accurate Jacobian approximation.

-4 an input value was found illegal by the GEARB package. An error message is printed on tape 3.

-5 insufficient storage available. Increase array dimensions in subroutine MOL1D

-6 illegal input NPTS < 5 or NPDE < 1

-7 illegal input KEQN ≠ ±1,±2,±3,-4

-8 illegal input KBC ≠ 1,2,3

-9 illegal input MODE ≠ 0, 1, 2

-10 illegal input MJAC ≠ 0,1,2,3,4,5

-11 input error in MORD

-12 illegal input MOUT > 6

-13 error in TOUT array, TLAST never reached

-14 error in TOUT array, TOUT(I + 1) $\leq$ TOUT(I) for some I

-15 error in plotting parameters in COMMON/MOLPLT/

-16 The Fourier method can be used only if there are periodic boundaries and NPTS=2^N for some N. This is a temporary restriction which will be removed in later versions.

-17 XM is not evenly spaced

If the solution is to be plotted (i.e. MOUT > 2) then the following common block must be provided by the user:

```
COMMON/MOLPLT/NPLT,IDEN,MG,XAL,UAL,XAXIS(2),UAXIS(NPDE,2)
```

with the following values and meanings for its arguments.

NPLT = indicates how the solution is to be plotted

1 each PDE component is plotted on a separate set of axes

2 all the PDE components are plotted on the same set of axes at each output time

3 all the PDE components are plotted on the same axes for the entire run

IDEN = plot identification label (as read with an A10 format)

MG = graph method parameter to determine the type of line used to plot the solution. If MG > 0 then the solution data points are designated by x's,

otherwise an unmarked line is drawn. MG also indicates if quadratic interpolation is to be used between the data points. It has the following values and meanings:

$\pm$1 no interpolation

$\pm$2 interpolate one point between each pair of data points

$\pm$3 interpolate two points between each pair of data points

XAL = length of x axis in inches ($\leq$ 10)

UAL = length of y axis in inches ($\leq$ 10)

XAXIS(1) = minimum value of x axis

XAXIS(2) = maximum value of x axis

UAXIS(I,1)=minimum value of U(I) axis

UAXIS(I,2)=maximum value of U(I) axis. To prevent U(I) from being plotted, set UAXIS(I,2) = UAXIS(I,1).

Additional communication between the user and the package can be gained through the common block:

```
COMMON/MOLINF/DTL,DTAVG,LORD,ORDAVG,NSTEPS,NPDEE,NJACE,CPTIME
```

with the following meanings and values for the arguments:

DTL = last time step size

DTAVG = average time step in the calculation

LORD = the order of the time integration method last used

ORDAVG = average order of the time integator used in the calculation

NSTEPS = the cumulative number of time steps taken

NPDEE = the cumulative number of PDE evaluations

NJACE = the cumulative number of Jacobian evaluations of the differential difference equation

CPTIME = the cumulative computer running time used for the integration

Approximations of the spatial derivatives of the solution at the last PDE evaluation are contained in the following common blocks:

```
COMMON / MOLC2 / UX(NPDE, NPTS)
COMMON / MOLC3 / UXX(NPDE, NPTS)
COMMON / MOLC4 / FX(NPDE, NPTS) .
```

As mentioned in Section 4 the accuracy of the numerical solution can be found in calculations where the exact solution is known by calling subroutine COMPARE. The parameter list and calling sequence for this subroutine is:

```
SUBROUTINE COMPARE(N,U,V,VS,UL2,VL2,DL1,DL2,DMAX,DPH,
                   CDL1,CDL2,CDMAX,IND)
```

with the following values and meanings for the calling parameters:

N = input parameter equal to the number of spatial mesh points

U = input array of length N

V = input array of length N to be compared with U

VS = output array of length N equal to the input vector V shifted to correct for the phase error in the dominate Fourier mode of U

UL2 = L_2 norm of U

VL2 = L_2 norm of V

DL1 = difference between U and V in the L_1 norm

DL2 = difference between U and V in the L_2 norm

DMAX = difference between U and V in the maximum (L_∞) norm

DPH = phase difference in the dominant Fourier mode of U

CDL1 = the difference between U and VS in the L_1 norm

CDL2 = the difference between U and VS in the L_2 norm

CDMAX = the difference between U and VS in the maximum (L_∞) norm

IND = indicator used for both input and output. On input IND > 0 flags no printed output. If IND < 0 then the differences between U and V are printed on tape 3. The magnitude of IND has the following values and meanings:

$\pm$1 do not compute phase differences

$\pm$2 compute phase differences.

IND returns zero if an error was encountered, otherwise it returns unchanged.

To structure MOL1D into an operational program the user must provide a main program and three subroutines. In the main program the arguments of the package are initialized and subroutine MOL1D is called. The subroutines describe the PDE system to be solved. For example, consider the following system of equations.

$$(u_1)_t = (u_1)_{xx} + f_1(u)_x = (u_1)_{xx} + \frac{1}{2}(u_1^2)_x$$

$$(u_2)_t = (u_2)_{xx} - f_2(u)_x = (u_2)_{xx} - \frac{1}{2}(u_2^2)_x$$

with the boundary conditions

$$a_{\ell_1} u_1(0,t) + b_{\ell_1} u_1(0,t)_x = c_{\ell_1}$$

$$a_{\ell_2} u_2(0,t) + b_{\ell_2} u_2(0,t)_x = c_{\ell_2}$$

and

$$a_{r_1} u_1(1,t) + b_{r_1} u_1(1,t)_x = c_{r_1}$$

$$a_{r_2} u_2(1,t) + b_{r_2} u_2(1,t)_x = c_{r_2}$$

where

$$a_{\ell_i} = a_{r_i} = c_{\ell_i} = c_{r_i} = 0 \quad , \quad i = 1,2,$$

$$b_{\ell_i} = b_{r_i} = 1 \ .$$

The subroutines supplied by the user for this problem are:

```
SUBROUTINE PDE(UT,U,UX,UXX,FX,T,X,IX,NPDE)

DIMENSION UT(NPDE),U(NPDE),UX(NPDE),UXX(NPDE),FX(NPDE)

      This subroutine defines the time derivative of
      the ith  PDE component UT(I) as a function of
      X,T,U,UX,UXX and the derivative of f, FX.  The
      spatial point X is the IX-th element in the array
      of mesh points XM defined in the main program.

UT(1) = UXX(1) + FX(1)

UT(2) = UXX(2) - FX(2)

RETURN

END


SUBROUTINE FUNC(F,U,UX,UXX,T,X,IX,NPDE)

DIMENSION F(NPDE), U(NPDE), UX(NPDE), UXX(NPDE)

      This subroutine defines f as a function of T,X,U,UX, and UXX.

F(1) = 0.5 * U(1) * U(1)

F(2) = 0.5 * U(2) * U(2)

RETURN

END


SUBROUTINE BNDRY(T,UL,AL,BL,CL,UR,AR,BR,CR,NPDE)

DIMENSION UL(NPDE),AL(NPDE),BL(NPDE),CL(NPDE),UR(NPDE),

  AR(NPDE),BR(NPDE),CR(NPDE)

      This subroutine defines the coefficients a_i, b_i and c_i
      for the left and right boundaries of eq. (A.2).  UL  and UR
      contain the solution at these boundaries at time T and shouldn't
      be altered.  If b_i ≠ 0 then a_i, b_i and c_i  can be functions
      of u and t, otherwise they can be functions of t only.
      If no boundary conditions are imposed, there is a free
      boundary or the boundary is defined by an ODE in subroutine
      PDE, then the coefficients should be set to zero.
```

```
AL(1) = 0.0
BL(1) = 1.0
CL(1) = 0.0      } left boundary
AL(2) = 0.0
BL(2) = 1.0
CL(2) = 0.0

AR(1) = 0.0
BR(1) = 1.0
CR(1) = 0.0      } right boundary
AR(2) = 0.0
BR(2) = 1.0
CR(2) = 0.0

RETURN
END
```

We now list an executable program to solve the KdV equation

$$u_t + uu_x + 0.03 \quad u_{xxx} = 0 \ , \quad 0 \leq x \leq 2\pi$$

with the initial conditions

$$u(x,0) = \sin (x)$$

and periodic boundaries.

```
      PROGRAM MTEST(INPUT,OUTPUT,TAPE3=OUTPUT)
C
C     PROGRAM TO SOLVE KDV EQUATION
C
      DIMENSION UZ(40),XM(40),MORD(3),TOUT(3)
      COMMON /MOLPLT/ NPLT,IDEN,MG,XAL,UAL,XAXIS(2),UAXIS(2)
C
C     SET PARAMETERS FOR MOL1D
C
      NPDE=1
      NPTS = 40
      KEQN = -2
      KBC=1
      METH = 24
      EPS = 1.E-6
      MORD(1) = 0
      MORD(2) = 4
      MORD(3) = 4
      TINT=0.0
      TLAST = 3.
      MOUT=4
      TOUT(1) = 0.0
      TOUT(2) = 1.5
      TOUT(3) = 3.
      PI2 = 2.*ACOS(-1.)
      DX = PI2/NPTS
      DO 20 I=1,NPTS
      XM(I)=(I-1)*DX
      UZ(I) = SIN(XM(I))
   20 CONTINUE
      KMOL = 0
C
C     SET PLOTTING PARAMETERS
C
      NPLT = 1
      IDEN=10HKDV EQN.
      MG = -3
      XAL = 4.
      UAL = 4.
      XAXIS(1)=XM(1)
      XAXIS(2)=XM(NPTS)+DX
      UAXIS(1) = -1.
      UAXIS(2) = 2.
C
      CALL MOL1D(NPDE,NPTS,KEQN,KBC,METH,EPS,MORD,TINT,
     1 TLAST,MOUT,TOUT,UZ,XM,KMOL)
      CALL EXIT
      END
```

```
      SUBROUTINE PDE(UT,U,UX,UXX,FX,T,X,IX,NPDE)
      DIMENSION U(NPDE),UT(NPDE),UX(NPDE),UXX(NPDE),FX(NPDE)
C
      UT(1) = FX(1)
C
      RETURN
      END


      SUBROUTINE FUNC(F,U,UX,UXX,T,X,IX,NPDE)
      DIMENSION F(NPDE),U(NPDE),UX(NPDE),UXX(NPDE)
C
      F(1) = -0.5*U(1)*U(1) - 0.03*UXX(1)
C
      RETURN
      END


      SUBROUTINE BNDRY(T,UL,AL,BL,CL,UR,AR,BR,CR,NPDE)
C
C     MOL1D DOES NOT CALL THIS SUBROUTINE
C     IF THERE ARE PERIODIC BOUNDARIES
C
      RETURN
      END
```

OUTPUT;

```
THE INPUT PARAMETERS ARE:

NPDE =  1   NPTS = 40   KBC = 1
KEQN = -2   METH = 24   EPS = 1.0000E-06

MORD(1,1) = 0   MORD(1,2) = 4   MORD(1,3) = 4

TINT =  0.          TLAST =  3.0000E+00   MOUT = 4

TOUT =     0.        1.5000E+00 3.0000E+00

********** PLOTTING PARAMERERS **********

KDV EQN.   PLOTS
 200 MAXIMUM INCHES (OR 16MM FRAMES)
   1 PLOTS PER FRAME MG = -3 XAL = 4.00 UAL = 4.00

X    AXIS RANGES BETWEEN  0.          AND   6.2832E+00
U(1) AXIS RANGES BETWEEN -1.0000E+00 AND   2.0000E+00
```

********** TIME = 0. **********

DT = 0. MOL ORDER = 0
TIME STEPS = 0 DT EVALUATIONS = 0
CPU TIME = .205 SECONDS

I	X	U(I)	UT(I)
1	0.	0.	2.9999E-02
2	1.57E-01	1.5643E-01	-1.2483E-01
3	3.14E-01	3.0902E-01	-2.6527E-01
4	4.71E-01	4.5399E-01	-3.7765E-01
5	6.28E-01	5.8779E-01	-4.5111E-01
6	7.85E-01	7.0711E-01	-4.7863E-01
7	9.42E-01	8.0902E-01	-4.5774E-01
8	1.10E+00	8.9101E-01	-3.9076E-01
9	1.26E+00	9.5106E-01	-2.8453E-01
10	1.41E+00	9.8769E-01	-1.4977E-01
11	1.57E+00	1.0000E+00	3.2984E-14
12	1.73E+00	9.8769E-01	1.4977E-01
13	1.88E+00	9.5106E-01	2.8453E-01
14	2.04E+00	8.9101E-01	3.9076E-01
15	2.20E+00	8.0902E-01	4.5774E-01
16	2.36E+00	7.0711E-01	4.7863E-01
17	2.51E+00	5.8779E-01	4.5111E-01
18	2.67E+00	4.5399E-01	3.7765E-01
19	2.83E+00	3.0902E-01	2.6527E-01
20	2.98E+00	1.5643E-01	1.2483E-01
21	3.14E+00	2.4991E-14	-2.9999E-02
22	3.30E+00	-1.5643E-01	-1.8409E-01
23	3.46E+00	-3.0902E-01	-3.2236E-01
24	3.61E+00	-4.5399E-01	-4.3111E-01
25	3.77E+00	-5.8779E-01	-4.9965E-01
26	3.93E+00	-7.0711E-01	-5.2105E-01
27	4.08E+00	-8.0902E-01	-4.9301E-01
28	4.24E+00	-8.9101E-01	-4.1500E-01
29	4.40E+00	-9.5106E-01	-3.0307E-01
30	4.56E+00	-9.8769E-01	-1.5915E-01
31	4.71E+00	-1.0000E+00	-9.6126E-14
32	4.87E+00	-9.8769E-01	1.5915E-01
33	5.03E+00	-9.5106E-01	3.0307E-01
34	5.18E+00	-8.9101E-01	4.1500E-01
35	5.34E+00	-8.0902E-01	4.9301E-01
36	5.50E+00	-7.0711E-01	5.2105E-01
37	5.65E+00	-5.8779E-01	4.9965E-01
38	5.81E+00	-4.5399E-01	4.3111E-01
39	5.97E+00	-3.0902E-01	3.2236E-01
40	6.13E+00	-1.5643E-01	1.8409E-01

********** TIME = 1.5000E+03 **********

DT = 2.9366E-02 MOL ORDER = 4
TIME STEPS = 61 DT EVALUATIONS = 134
CPU TIME = 1.252 SECONDS

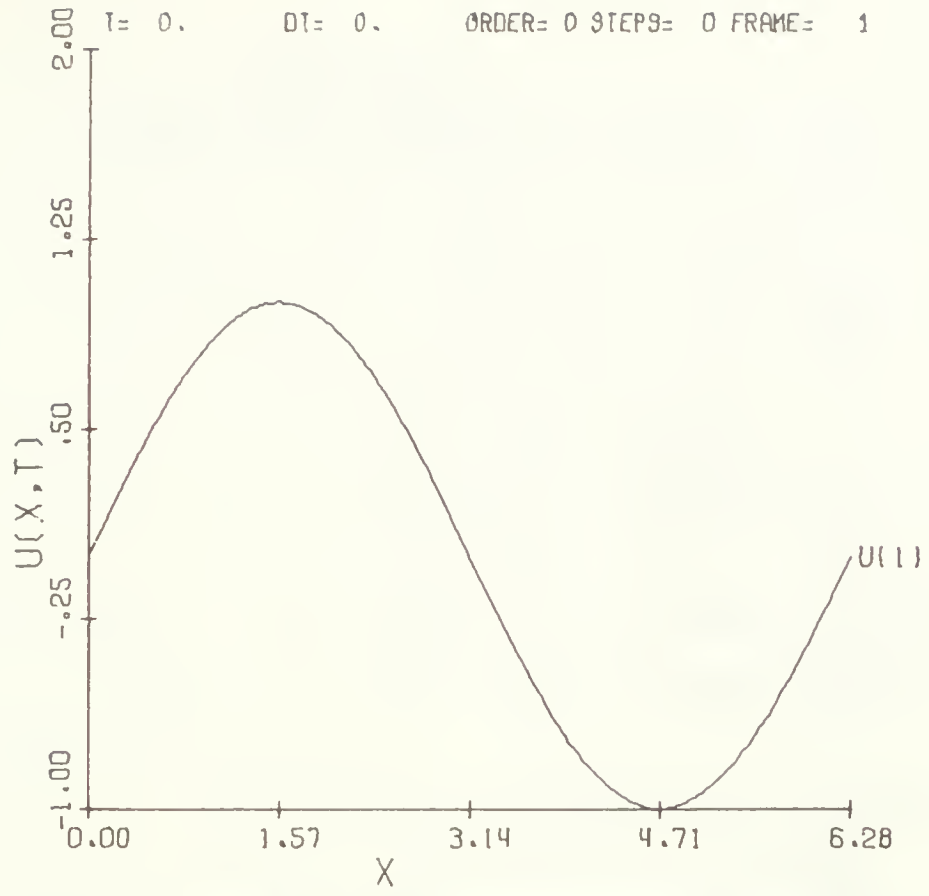
I	X	U(1)	UT(1)
1	0.	-1.9602E-03	1.4689E-01
2	1.57E-01	1.0047E-01	2.8075E-01
3	3.14E-01	1.9398E-01	1.8449E-01
4	4.71E-01	2.3923E-01	-1.6786E-01
5	6.28E-01	2.2203E-01	-5.6983E-01
6	7.85E-01	1.9009E-01	-8.9434E-01
7	9.42E-01	2.2690E-01	-4.0834E-01
8	1.10E+00	3.9900E-01	1.9271E-01
9	1.26E+00	6.7838E-01	7.0067E-01
10	1.41E+00	8.9413E-01	5.5456E-01
11	1.57E+00	8.3537E-01	-3.1960E-01
12	1.73E+00	5.1858E-01	-1.1121E+00
13	1.88E+00	1.9755E-01	-1.3400E+00
14	2.04E+00	1.2417E-01	-1.1769E+00
15	2.20E+00	4.6942E-01	-8.1634E-01
16	2.36E+00	1.2517E+00	-2.0097E-01
17	2.51E+00	1.9944E+00	5.9451E-01
18	2.67E+00	1.8002E+00	7.8386E-01
19	2.83E+00	8.3140E-01	3.9005E-01
20	2.98E+00	-4.3670E-02	1.0322E-01
21	3.14E+00	-5.4919E-01	4.6642E-02
22	3.30E+00	-7.8203E-01	8.3083E-02
23	3.46E+00	-8.7150E-01	1.4142E-01
24	3.61E+00	-8.9043E-01	1.8245E-01
25	3.77E+00	-8.7721E-01	2.1363E-01
26	3.93E+00	-8.4592E-01	2.2861E-01
27	4.08E+00	-8.0631E-01	2.4337E-01
28	4.24E+00	-7.5920E-01	2.4851E-01
29	4.40E+00	-7.0926E-01	2.4696E-01
30	4.56E+00	-6.5546E-01	2.2795E-01
31	4.71E+00	-6.0201E-01	1.9937E-01
32	4.87E+00	-5.4640E-01	1.7451E-01
33	5.03E+00	-4.9024E-01	1.6975E-01
34	5.18E+00	-4.2675E-01	1.8726E-01
35	5.34E+00	-3.6443E-01	1.9956E-01
36	5.50E+00	-2.9730E-01	1.7025E-01
37	5.65E+00	-2.3785E-01	7.6013E-02
38	5.81E+00	-1.8581E-01	-3.8923E-02
39	5.97E+00	-1.3944E-01	-9.7139E-02
40	6.13E+00	-8.2057E-02	-2.3337E-02

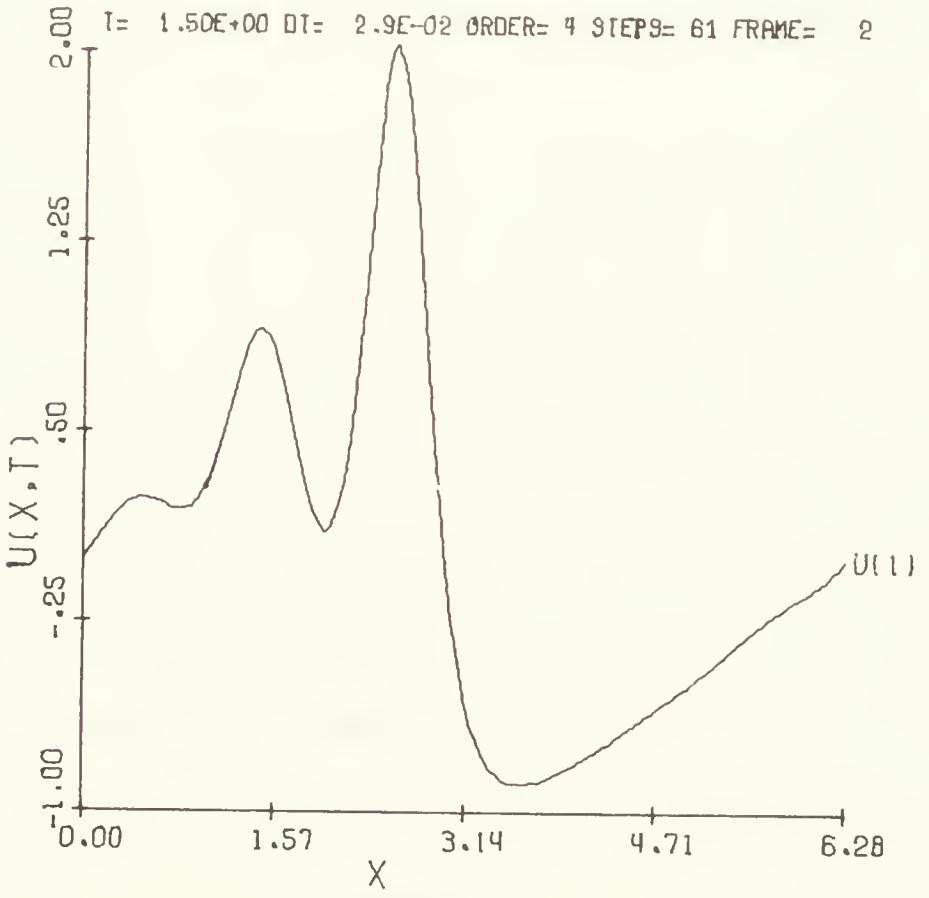
********** TIME = 3.0003E+01 **********

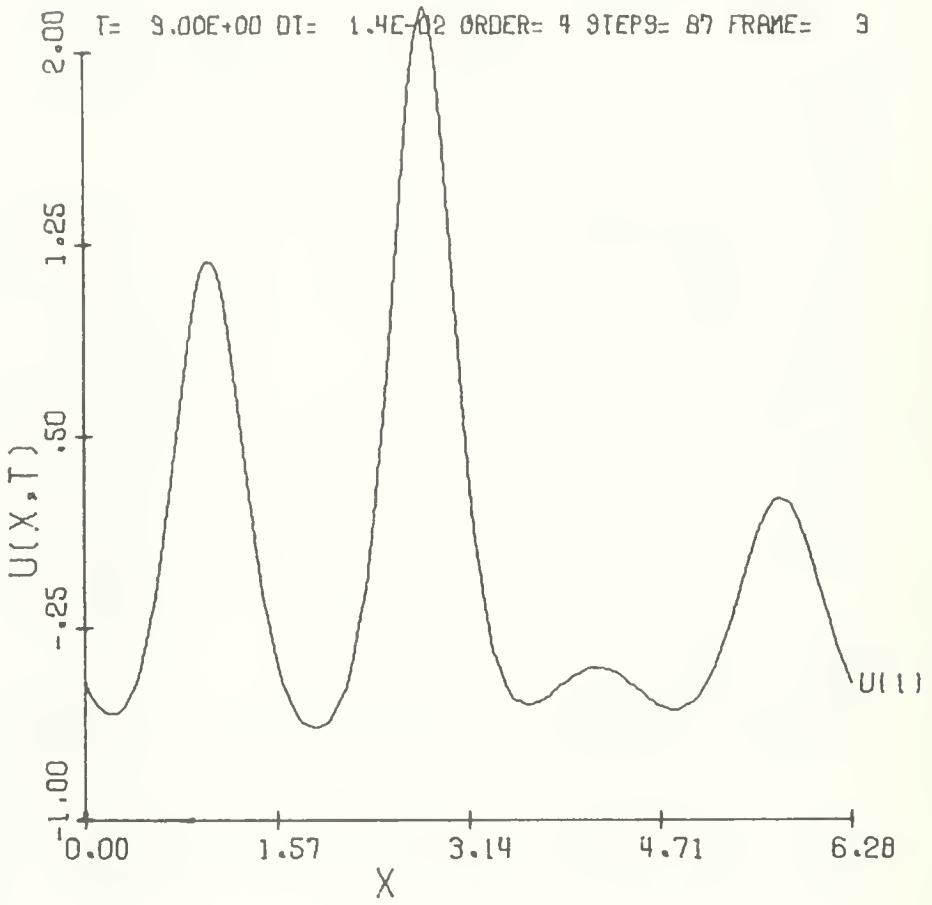
DT = 1.3063E-02 NOL ORDER = 4
TIME STEPS = 87 DT EVALUATIONS = 156
CPU TIME = 1.348 SECONDS

I	X	U(1)	UT(1)
1	0.	-4.7057E-01	-7.3735E-01
2	1.57E-01	-5.8143E-01	-3.8570E-01
3	3.14E-01	-5.6207E-01	-2.7762E-02
4	4.71E-01	-3.7244E-01	3.7811E-01
5	6.28E-01	3.1527E-02	7.8811E-01
6	7.85E-01	6.2140E-01	9.6612E-01
7	9.42E-01	1.1279E+00	4.3123E-01
8	1.10E+00	1.1271E+00	-5.1708E-01
9	1.26E+00	6.2412E-01	-9.1230E-01
10	1.41E+00	3.9349E-02	-6.5926E-01
11	1.57E+00	-3.6873E-01	-3.0129E-01
12	1.73E+00	-5.7874E-01	-6.0975E-02
13	1.88E+00	-6.4235E-01	-1.0675E-02
14	2.04E+00	-5.8573E-01	-1.1107E-01
15	2.20E+00	-3.7304E-01	-4.0570E-01
16	2.36E+00	1.0256E-01	-8.6824E-01
17	2.51E+00	9.4669E-01	-1.3294E+00
18	2.67E+00	1.9139E+00	-9.8245E-01
19	2.83E+00	2.1393E+00	4.6792E-01
20	2.98E+00	1.3306E+00	1.2492E+00
21	3.14E+00	3.7970E-01	1.0221E+00
22	3.30E+00	-2.1474E-01	6.8021E-01
23	3.46E+00	-4.8077E-01	5.4425E-01
24	3.61E+00	-5.5075E-01	5.4406E-01
25	3.77E+00	-5.2760E-01	5.5780E-01
26	3.93E+00	-4.6718E-01	4.5509E-01
27	4.08E+00	-4.1860E-01	2.1810E-01
28	4.24E+00	-4.0534E-01	-9.4509E-02
29	4.40E+00	-4.3936E-01	-3.8106E-01
30	4.56E+00	-4.9897E-01	-4.3940E-01
31	4.71E+00	-5.5012E-01	-3.9495E-01
32	4.87E+00	-5.7077E-01	-1.3273E-01
33	5.03E+00	-5.1790E-01	2.6091E-01
34	5.18E+00	-3.7318E-01	6.9309E-01
35	5.34E+00	-1.4952E-01	9.7387E-01
36	5.50E+00	9.9203E-02	8.7452E-01
37	5.65E+00	2.5100E-01	2.5137E-01
38	5.81E+00	2.1525E-01	-5.4788E-01
39	5.97E+00	6.1468E-03	-1.0121E+00
40	6.13E+00	-2.5608E-01	-9.9711E-01

TOTAL TIME STEPS = 148 AVERAGE TIME STEP = 2.0358E-02
TOTAL DT EVALUATIONS = 270 AVERAGE METHOD ORDER = 3.851
TOTAL CPU TIME = 2.90 SECONDS


T= 0. DT= 0. ORDER= 0 STEPS= 0 FRAME= 1
2.00
1.25
.50
-.25
-1.00
U(X,T)
U(1)
0.00
1.57
3.14
4.71
6.28
X


T= 1.50E+00 DT= 2.9E-02 ORDER= 4 STEPS= 61 FRAME= 2
2.00
1.25
.50
-.25
-1.00
U(X,T)
U(1)
0.00
1.57
3.14
4.71
6.28
X


T= 9.00E+00 DT= 1.4E-02 ORDER= 4 STEPS= 87 FRAME= 3
2.00
1.25
.50
-.25
-1.00
U(X,T)
0.00
1.57
3.14
4.71
6.28
X
U( )

Bibliography

[1] Abrashin, V. N., "Convergence and Error Estimates for the Straight-Line Method for Certain Problems of Mathematical Physics," Differentsial nye Uravneniya, Vol. 4, No. 7, 1968, pp. 1137-1332.

[2] Browning, G., Kreiss, H.-O., and Oliger, J., "Mesh Refinement," Mathematics of Computation, Vol. 27, No. 121, January 1973, pp. 29-39.

[3] Cardenas, A. F , and Karplus, W. J., "PDEL-A Language for Partial Differential Equations," Comm. ACM, Vol. 13, March 1970, pp.184-191.

[4] Elvius, T., and Sundström, "Computationally Efficient Schemes and Boundary Conditions for a Fine-Mesh Barotropic Model Based on the Shallow-Water Equations," Tellus, 25, 1973, pp. 132-156.

[5] Endquist, B., and Smedsaas, T., "DCG - A System for Atuomatic Code Generation for Hyperbolic Problems," SIGNUM Newsletter, December 1975.

[6] Fornberg, B., "On High Order Approximations of Hyperbolic Partial Differential Equations by a Fourier Method," Uppsala University Computer Science Report NR 39, April 1972.

[7] Fornberg, B., "On the Instability of Leap-Frog and Crank Nicholson Approximations of a Nonlinear Partial Differential Equation," Math. Comp., Vol. 27, 1973, pp. 47-57.

[8] Gary, J., "On the Optimal Time Step and Computational Efficiency of Difference Schemes for ODE," Journal of Computational Physics, Vol. 16, 1974, pp. 298-303.

[9] Gary, J., "Boundary Conditions for the Method of Lines Applied to Hyperbolic Systems," Technical Report No. CU-CS-073-75, Department of Computer Science, University of Colorado, July 1975.

[10] Gary, J., "The Method of Lines Applied to a Simple Hyperbolic Equation," Preprint, Computer Science Department, University of Colorado, 1975.

[11] Gekeler, E., "Linear Multistep Methods and Galerkin Procedures for Initial Boundary Value Problems," SIAM J. Numer. Anal., Vol. 13, No. 4, Sept. 1976, pp. 536-548.

[12] Gottlieb, D., and Turkel, E., "Dissipative Two-Four Methods for Time Dependent Problems," ICASE Report, 75-22, October 30, 1975.

[13] Gustafsson, B., "The Convergence Rate for Difference Approximations to Mixed Initial Boundary Value Problems," Math. Comp., Vol. 29, 1975, pp. 396-406.

[14] Harten, A., "The Method of Artificial Compression: I. Shocks and Contact Discontinuities," Courant Inst. Math. Sci., New York Univ., Report No. COO-3077-50, June 1974.

[15] Helgason, R., and Gary, J., "PDELAN User's Manual," National Center for Atmospheric Research, Boulder, Col., 1971.

[16] Hicks, J. S., and Wei, J., "Numerical Solution of Parabolic Partial Differential Equations with Two-Point Boundary Conditions by Use of the Methods of Lines," Journal of the Association for Computing Machinery, Vol. 14, No. 3, July 1967, pp. 549-562.

[17] Hindmarsh, A. C., "GEAR: Ordinary Differential Equation System Solver," Lawrence Livermore Laboratory Report UCID-30001, Rev. 3, 1974.

[18] Hindmarsh, A. C., "GEARB: Solution of Ordinary Differential Equations Having Banded Jacobian," Lawrence Livermore Laboratory Report UCID-30059, Rev. 1, March 1975.

[19] Hull, T. E., "The Validation and Comparison of Programs for Stiff Systems," Stiff Differential Systems, ed. R. A. Willoughby, Plenum Press, New York, 1974, pp. 151-164.

[20] Hull, T. E., "Numerical Solutions of Initial Value Problems for Ordinary Differential Equations," Numerical Solutions of Boundary Value Problems for Ordinary Differential Equations, ed. Aziz, A. K. Academic Press, 1975, pp. 3-26.

[21] Hull, T. E., Enright, W. H., Fellen, B. M., and Sedgwick, A. E., "Comparing Numerical Methods for Ordinary Differential Equations," SIAM J. Numer. Anal., Vol. 9, No. 4, Dec. 1972, pp. 603-637.

[22] Liskovets, O. A., "The Method of Lines," (Review), English Translation appeared in Differential Equations, Vol. 1, 1965, pp. 1308-1323.

[23] Kashef, B., and Bellman, R., "Solution of the Partial Differential Equation of the Hodgkin-Huxley Model Using Differential Quadrature," Mathematical Biosciences 19, 1974, pp. 1-8.

[24] Kreiss, H., and Oliger, J., "Methods for the Approximate Solution of Time Dependent Problems," GARP Publication Series, No. 10, February 1973.

[25] Kreiss, H. O., "Initial Boundary Value Problems for Hyperbolic Systems," Comm. Pure Appl. Math., Vol. 23, 1970, pp. 277-298.

[26] Kreiss, H. O., and Oliger, J.,"Comparison of Accurate Methods for the Integration of Hyperbolic Equations," Tellus 24, 1972, pp. 199-215.

[27] Krogh, F. T.,"VODQ/SVDQ/DVDQ -- Variable Order Integrators for the Numerical Solution of Ordinary Differential Equations," Jet Propulsion Laboratory Section 14, Subroutine Writeup, Pasadena, Cal., 1969.

[28] Lax, P. and Wendroff, B., "Systems of Conservation Laws," Comm. Pure Appl. Math., Vol. 13, 1960, pp. 217-237.

[29] Lax, P., "On the Stability of Difference Approximations to Solutions of Hyperbolic Equations with Variable Coefficients," Comm. Pure Appl. Math., Vol. 14, 1961, pp. 497-520.

[30] Loeb, A. M., and Schiesser, W. E., "Stiffness and Accuracy in the Method of Lines Integration of Partial Differential Equations," Stiff Differential Systems, ed. R. A. Willoughby, Plenum Press, New York, 1974, pp. 229-243.

[31] Madsen, N. K., and Sincovec, R. F.,"The Numerical Method of Lines for the Solution of Nonlinear Partial Differential Equations," Lawrence Livermore Laboratory UCRL-75142, September 1973.

[32] Madsen, N. K., and Sincovec, R. F., "PDEPACK: A New Tool for Simulation," Proceedings of the 1975 Summer Computer Simulation Conference, San Francisco, California, July 1975.

[33] Madsen, N. K., and Sincovec, R. F., "PDECOL: General Collocation Software for Partial Differential Equations," Lawrence Livermore Laboratory, Preprint, UCRL-78263, May 1976.

[34] Roberts, S. M., "On the Closed Form Solution of Troesch's Problem, J. Comp. Physics 21, 1976, pp. 291-304.

[35] Robertson, H. H., "The Solution of a Set of Reaction Rate Equations," in Numerical Analysis: An Introduction, ed. J. Walsh, Thompson Book. Co., Washington, D. C., 1966, pp. 178-182.

[36] Schiesser, W. E., LEANS III Introductory Programming Manual, Lehigh University Computing Center, Bethlehem, Pa., 1971.

[37] Shampine, L. F., and Gordon, M. K., Computer Solution of Ordinary Differential Equations: The Initial Value Problem, W. H. Freeman and Co., San Francisco, 1975.

[38] Shampine, L. F., Watts, H. A., and Davenport, S. M., "Solving Nonstiff Ordinary Differential Equations: The State of the Art," SIAM Review Vol. 18, No. 3, July 1976, pp. 376-411.

[39] Sincovec, R. F., and Madsen, N. K., "Software for Nonlinear Partial Differential Equations," ACM Transactions on Mathematical Software, Vol. 1, No. 3, September 1975, pp. 232-260.

[40] Sköllermo, G., "Error Analysis for the Mixed Initial Boundary Value Problem for Hyperbolic Equations," Uppsala University, Department of Computer Sciences, Report No. 63, December 1975.

[41] Swartz, B.K., "The Construction and Comparison of Finite Difference Analogs of Some Finite Element Schemes," Mathematical Aspects of Finite Elements in Partial Differential Equations, ed. Boor, C. de, Academic Press, 1974, pp. 279-312.

[42] Swartz, B., and Wendroff, B., "The Relative Efficiency of Finite Difference and Finite Element Methods, I: Hyperbolic Problems and Splines," SIAM J. Numer. Anal. Vol. 11, No. 5, October 1974, pp. 979-993.

[43] Swartz, R., and Wendroff, B., "The Comparative Efficiency of Certain Finite Element and Finite Difference Methods for a Hyperbolic Problem." Conference on the Numerical Solution of Differential Equations, ed. G.A. Watson, Lecture Notes in Mathematics, No. 363, Springer-Verlag, 1974, pp. 153-163.

[44] Thompson, R. C., "Convergence and Error Estimates for the Method of Lines for Certain Nonlinear Elliptic and Elliptic-Parabolic Equations," SIAM J. Numer. Anal. Vol. 13, No. 1, March 1976, pp. 27-43.

[45] Troesch, B. A., "A Simple Approach to a Sensitive Two-Point Boundary Value Problem," J. of Comp. Physics 21, 1976, pp. 279-290.

[46] Voigt, A., "Line Method Approximations to Cauchy Problem for Nonlinear Parabolic Differential Equations," Numer. Math. 23, 1974, pp. 23-36.

[47] Walter, W.,"Differential and Integral Inequalities," Springer-Verlag, Berlin-Heidelberg-New York, 1970, pp. 275-303.

[48] Willoughby, R. A., Stiff Differential Systems, Plenum Press, New York, 1974.

[49] Zellner, M. G., "DSS - Distributed System Simulator," Ph.D. Dissertation, Lehigh University, Bethlehem, Pa., June 1970.

[50] Varah, J. M., "On the Stability of Boundary Conditions for Separable Difference Approximations to Parabolic Equations," University of British Columbia, Dept. of Computer Science, Technical Report 76-4, June 1976.

This report was prepared as an account of Government sponsored work. Neither the United States, nor the Administration, nor any person acting on behalf of the Administration:

A. Makes any warranty or representation, express or implied, with respect to the accuracy, completeness, or usefulness of the information contained in this report, or that the use of any information, apparatus, method, or process disclosed in this report may not infringe privately owned rights; or

B. Assumes any liabilities with respect to the use of, or for damages resulting from the use of any information, apparatus, method, or process disclosed in this report.

As used in the above, "person acting on behalf of the Administration" includes any employee or contractor of the Administration, or employee of such contractor, to the extent that such employee or contractor of the Administration, or employee of such contractor prepares, disseminates, or provides access to, any information pursuant to his employment or contract with the Administration, or his employment with such contractor.

This book may be kept JAN 24 1977
FOURTEEN DAYS
A fine will be charged for each day the book is kept overtime.

DEC 16 1976
JAN - 7 1977
JAN 17 1979
FEB 1 1979
NOV 12 1979
AUG 13 1981
FEB 21 1984
ILL - 4.13.89
ILL - 1.26.92

GAYLORD 142

NYU COO-3077-139 c.1

Hyman

The method of lines solution of part. diff. equations.

N.Y.U. Courant Institute of Mathematical Sciences
251 Mercer St.
New York, NY 10012